Technische Hochschule Köln
University of Applied Sciences
Campus Gummersbach
Fakultät für Informatik und Ingenieurwesen

Fachhochschule Dortmund
University of Applied Sciences and Arts
Fachbereich Informatik

Verbundstudiengang Wirtschaftsinformatik

Abschlussarbeit

zur Erlangung des Mastergrades

Master of Science

in der Fachrichtung Informatik

Ortsbasierte Indoor Optical-See-Through Augmented Reality mit der Google Glass

Erstprüfer:	Prof. Dr. Faeskorn-Woyke
Zweitprüfer:	Prof. Dr. Karsch
vorgelegt am:	xx.xx.2015
von cand.:	Jonathan Scheinmann
aus:	Talstr. 37 40878 Ratingen
Tel.:	0152 / 21538174
Email:	jscheinmann@googlemail.com
Matr.-Nr.:	8087337

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

AOA ..... Angle of Arrival
API ..... Application Programming Interface
AR ..... Augmented Reality
BLE ..... Bluetooth Low Energy
COO ..... Cell of Origin
DAO ..... Data Access Object
DTO ..... Data Transfer Object
FOV ..... Field of View
GDK ..... Glass Development Kit
GPS ..... Global Positioning System
GUI ..... Graphical User Interface
HMD ..... Head-Mounted-Display
HUD ..... Head-Up-Display
LLA ..... Latitude, Longitude, Altitude
PC ..... Personal Computer
PDA ..... Personal Digital Assistant
POI ..... Point of Interest
REST ..... Representational State Transfer
RSSI ..... Received Signal Strength Indicator
RTS ..... Request to Send
SDK ..... Software Development Kit
SIFT ..... Scale-invariant feature transform
TDOA ..... Time Difference of Arrival
TOA ..... Time of Arrival

1 Einleitung

1.1 Motivation

> *„A display connected to a digital computer gives us a chance to gain familiarity with concepts not realizable in the physical world. It is a looking glass into a mathematical wonderland.“*
>
> -Ivan Sutherland (1965)

Zu den Meilensteinen der Informatik-Forschung zählt sicherlich das von Ivan Sutherland 1968 entwickelte, erste funktionsfähige Head-Mounted-Display (HMD). Es ermöglichte mit einem semi-transparenten Display die real wahrgenommene Welt um virtuell projizierte Objekte zu erweitern, die auf Veränderungen der Kopfposition ihre Lage veränderten. Das von ihm entwickelte Optical-See-Through HMD gilt daher als Wegbereiter dessen was heute unter dem Begriff Augmented Reality (AR) bekannt ist. Der Apparat war allerdings so schwer, dass er von der Decke getragen werden musste. Der Rechner hatte die Größe eines Wohnzimmerschrankes. Aufgrund dieser Restriktionen verweilte Augmented Reality lange Zeit in Forschungslaboren, die Praxistauglichkeit war schlichtweg nicht gegeben.
In den darauf folgenden Jahren machte die Rechnertechnik enorme Fortschritte. Computer-Hardware wurde kleiner und leistungsfähiger. So brachten vor allem mobile Geräte wie Smartphones und Tablets, die notwendigen Voraussetzungen mit um Augmented Reality zu ermöglichen, wie etwa eine Kamera, Intertialsensorik oder GPS.
Stand heute jedoch, scheint sich die Vision Sutherland's, die reale Welt in ein mathematisches Wunderland zu verwandeln nicht erfüllt zu haben. Zwar finden sich zahlreiche AR Anwendungen im mobilen Bereich wie Ortsbasierte Apps, die Umgebungsinformationen ortsabhängig einblenden oder Marker-basierte Apps, die in Zeitschriften für Werbezwecke eingesetzt werden. Ein wichtiger Aspekt der AR bleibt jedoch aus: Die gänzliche Verschmelzung der Realität mit der Virtualität. Dies ist zum einen auf die technischen Herausforderungen zurückzuführen. So können Trackingkomponenten noch nicht die erforderlichen Antwortzeiten erfüllen, Bewegungen wirken im Bild dadurch zeitlich versetzt, oder 3D-Bilder wirken aufgrund der falschen Lichteinwirkungen unecht etc.. Zum anderen liegt die Ursache aber auch in der Akzeptanz der Nutzer. Head-Mounted-Displays, die bereits speziell für die Industrie eingesetzt werden, wirken klobig und sind in der Bedienung nicht intuitiv genug, um sie im Alltag einzusetzen.
Im Jahr 2012, also 44 Jahre nachdem Sutherland das erste HMD vorstellte, wagt das Unternehmen Google mit der Google Glass den Versuch das

HMD als ein massentaugliches Produkt zu etablieren. Die Google Glass ist ein 43 Gramm leichtes Brillengestell, bestehend aus einem speziellen Rahmen mit Miniaturcomputer und einem Prisma zur Projektion des Bildes auf die Retina des Auges. Die Brille ermöglicht Optical-See-Through. Diese AR-Methode ermöglicht die reale Welt mit eigenen Augen wahrzunehmen, während ein optischer Kombinierer diese um virtuelle Bilder erweitert. Damit gilt sie auch als Aufschwung dessen, was Sutherland als Vision formulierte.
Google musste sich jedoch schnell Kritik aussetzen, die Brille sei nicht ausgereift, bzw. funktional sehr eingeschränkt und könne durch das Prisma im oberen rechten Sichtbereich nicht als Augmented Reality-Brille bezeichnet werden. Tatsächlich wurde die Brille nur für Test- und Entwicklungszwecke an ausgewählte Personen freigegeben, wodurch das Potenzial und der Nutzen der Brille bislang noch nicht vollständig ausgeschöpft werden konnte.
Zahlreiche Entwickler haben bereits erste Prototypen für die Datenbrille entwickelt, wie etwa Apps für Gesichtserkennung, Spiele oder nützliche Tools für den Alltag. Darunter auch einige ortsabhängige AR-Apps, die Umgebungsinformationen zu Gebäuden oder alten Besichtigungsstätten anzeigen können. Diese Art der AR wird auch ortsbasierte AR genannt und dient dazu kontextabhängige Informationen aus der aktuellen Position und dem Sichtfeld des Benutzers zu ermitteln. Im Indoor-Bereich gab es hierzu bisher jedoch wenige Ansätze, mit der Google Glass bisher gar keine.
Die folgende Arbeit stellt einen ortsbasierten AR Ansatz innerhalb von Gebäuden vor, bei dem die Google Glass als Optical-See-Through-Device zum Einsatz kommt. In Form einer Machbarkeitsstudie soll der Nutzen der Google Glass im Alltag und dessen Potenzial als AR-Brille anhand eines Fallbeispiels einer AR-Navigationsapplikation an der TH Köln aufgezeigt werden. Hierfür wird ein Konzept zum Aufbau eines geeigneten Umgebungsmodells vorgestellt und mögliche Verfahren zur Positionsbestimmung innerhalb von Gebäuden evaluiert. Anhand der Position sollen Kontext-abhängige virtuelle Objekte in das Sichtfeld des Benutzers eingeblendet werden.

1.2 Szenario und Problemstellung

Der Campus Gummersbach der Technische Hochschule besteht aus etwa 60 Lehrräumen, die sich in 3 Etagen erstrecken und kann circa 3200 Studenten aufnehmen. Für das Auffinden von Räumen innerhalb des Gebäudes stehen Raumpläne zur Verfügung, jedoch stellt sich oftmals die Suche nach dem richtigen Raum, besonders für Erstsemester-Studenten oder Besucher, als Herausforderung dar. Dies gilt insbesondere für die Suche nach freien Räumen, da diese als solche in der Regel nicht gekennzeichnet sind.
Aus dieser Problemstellung heraus wurde entschieden, eine ortsbasierte AR-

Navigations-App zu implementieren, die es Benutzern erlaubt innerhalb des Gebäudes zu navigieren. Als geeignetes Fallbeispiel für den Nutzen von Augmented Reality sollen dem Benutzer Kontextinformationen in seiner Umgebung, wie z. B. zu Räumlichkeiten angezeigt werden. Hierbei sollte zu einem Lehrraum die Raumnummer und die aktuelle und nächste Vorlesung erkennbar sein, sodass der Benutzer freie von besetzten Räumen unterscheiden kann. Des Weiteren sollte man von einer gewissen Ausgangsposition zu einem Raum bzw. Flur des gesuchten Raumes navigieren können.

1.3 Aufbau der Arbeit

Die Masterarbeit ist wie folgt strukturiert: Im zweiten Kapitel wird Augmented Reality definiert und dessen Grundbausteine erläutert. Des Weiteren wird die Google Glass vorgestellt und die grundlegenden Konzepte der Indoor-Navigation dargelegt. Das dritte Kapitel umfasst die Analyse-Phase, in welcher auf Basis der Anforderungen eine AR Methodik gewählt und damit die Grundvoraussetzungen beschrieben werden. Zugleich befasst sich das Kapitel mit dem Grundkonzept der Augmented Reality Applikation sowie mit der Evaluation der zu verwendeten Verfahren und Methoden. Kapitel 4 beschreibt die Implementierung, wobei die Gesamtarchitektur des Systems und dessen Hauptkomponenten erläutert werden. Im darauffolgenden Kapitel erfolgt die Evaluation anhand eines Testfalls. Die Arbeit schließt mit einem Fazit und einem Ausblick ab.

2 Grundlagen

Das folgende Kapitel gibt, basierend auf der Arbeit „Augmented Reality Definition, Realisierung und Nutzen“[1], einen Einblick in die Grundlagen der erweiterten Realität und führt fundamentale Definitionen und Begriffe ein, die für das Verständnis der Arbeit notwendig sind. Ferner werden wesentliche Konzepte und der Standortbestimmung beleuchtet und der Stand der Technik vorgestellt. Zuletzt wird die Google Glass eingeführt.

2.1 Augmented Reality

2.1.1 Definition

Der Begriff „Augmented Reality“, oder zu deutsch „erweiterte Realität“, wurde erstmalig in der Flugzeugindustrie verwendet. Die Boeing Mitarbeiter Tom Caudell und David Mizell entwickelten im Rahmen eines Forschungsprojektes ein Head-Up Display[2], um Flugzeugbauern das Verlegen von Kabelbäumen auf sogenannten Formboards zu vereinfachen. Die Head-Up Displays markierten auf den Formboards die Stellen, an denen sie die Kabel verlegen mussten. Auf diese Weise konnten Flugzeugbauer ihre Effizienz erheblich steigern.
Eine allgemein anerkannte Definition für Augmented Reality existiert zum jetzigen Stand nicht. Die meist zitierte Definition stammt von Ronald T. Azuma. In seiner 1997 erschienenen Schrift „A Survey of Augmented Reality“[3] führt dieser drei grundlegende Merkmale auf, die ein Augmented Reality-System auszeichnet.

1. **Combines Real And Virtual (Kombiniert Realität mit Virtualität)**
2. **Interactive In Real Time (In Echtzeit interaktiv)**
3. **Registered In Three Dimensions (Registrierung in drei Dimensionen)**

Das erste und auch das wichtigste Merkmal das Azuma beschreibt, charakterisiert das Grundprinzip von Augmented Reality: Die Realitätswahrnehmung des Menschen wird durch virtuelle Inhalte angereichert. Im Idealfall ist dabei das virtuelle Bild mit dem real wahrgenommenen Bild vollständig verschmolzen.
Das zweite Merkmal besagt, dass sich die virtuell angereicherten Inhalte

[1] Vgl. Scheinmann, *Augmented Reality - Definition, Realisierung und Nutzen*, S.5-S.22
[2] Vgl. Caudell und Mizell, Augmented Reality: An Application of Heads-Up Display Technology to Manual Manufacturing Processes, S.659-669
[3] Vgl. Azuma, Presence: Teleoperators and Virtual Environments, Nr. 4, Bd. 6, 1997, S.2

in Echtzeit, abhängig vom Kontext des Blickfeldes stets aktualisieren. Im Idealfall geschehe dies in einem Zeitraum, der geringer sei, als der zeitliche Abstand zum nächsten Bild[4], damit der Eindruck entstehe, die virtuellen Objekte seien real.
Das letzte Merkmal setzt voraus, dass sich die virtuell erzeugten Inhalte im dreidimensionalen Raum einfügen müssen.
Azuma's Definition nimmt wenig Bezug auf die technische Realisierung von Augmented Reality, sodass hier ein größerer Interpretationsspielraum entsteht. So könnte man sowohl Head-Mounted-Displays, die am Kopf getragen werden, wie auch weitere Techniken wie Fernseher, Smartphones oder Tablets zur Realisierung von AR genutzt werden. Als Beispiel wäre hier die bei einer Freistoßsituation angezeigte Entfernung von einem Fußballspieler zum Tor bei einer Fußballübertragung im Fernsehen zu nennen. Hierbei wird nämlich das Live-Bild um virtuelle Inhalte wie bspw. ein Pfeil und die entsprechende Entfernung erweitert. Prinzipiell würden damit alle Voraussetzungen von Azuma erfüllt werden. Handelt es sich wiederum um eine angezeigte Abseitslinie in der Wiederholung, so würde dies dem zweiten Kriterium, nämlich der Echtzeitüberlagerung widersprechen, da die aufgenommenen Kamerabilder nachbearbeitet wurden. Da Echtzeit in Azuma's Definition jedoch nicht spezifiziert wird, könnte genauso das Freistoß-Szenario mit Live-Bild Übertragung dem Kriterium widersprechen, da eine Live-Übertragung immer zeitversetzt beim Empfänger ankommt.
Das letzte Kriterium muss in der Praxis nicht zwangsläufig erfüllt sein. Hier gibt es durchaus Anwendungsfälle, in denen die Registrierung eine untergeordnete Rolle spielt und vielmehr der Bezug zum dreidimensionalen Raum relevant ist. Zum Beispiel wird eine App, die beim Blick auf den Himmel, je nach Blickrichtung jeweilige Informationen zu Sternenbilder darstellt, auch als AR App bezeichnet.

Schließlich stellt sich bei Azuma's Definition die Frage, wie viel Virtualität die Realitätswahrnehmung zulässt, d.h. ab welchem Grad der Virtualität noch von erweiterter Realität gesprochen werden kann. Es ist daher eine klare Abgrenzung zur Virtuellen Realität notwendig, bei welcher die Wahrnehmung der Realität in einer in Echtzeit virtuellen Umgebung stattfindet. Paul Milgram und Fumio Kishino haben sich dieser Problematik angenommen und 1994 das sogenannte Realitäts-Virtualitäts-Kontinuum definiert[5].

Die Abbildung 1 stellt das Realitäts-Virtualitäts-Kontinuum grafisch

[4] Vgl. Tönnis, *Augmented Reality*, S.2
[5] Vgl. Milgram und Kishino, IEICE Trans. Information Systems, Nr. 12, Bd. E77–D, 1994, S.283
[7] A. a. O.

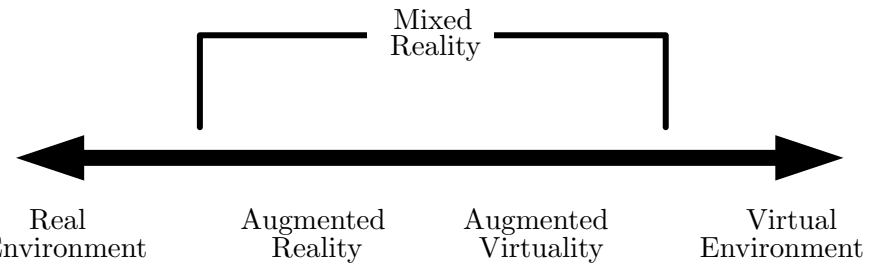

Abbildung 1: Das Realitäts-Virtualitäts-Kontinuum. Eigene Darstellung in Anlehnung an Milgram und Kishino[7]

dar. Es erstreckt sich über die beiden Endpunkte der Realität und Virtualität und illustriert die dazwischen liegenden Übergänge. Man kann sich das Kontinuum also als einen Raum aus Definitionsmengen vorstellen, wobei die Grenzen eher fließend ineinander überlaufen. Nach links hin wächst der Grad der Realität, während nach rechts hin der Grad die Virtualität steigt. Wiegt der Teil der Realität über, spricht man von Augmented Reality. Wiegt der Teil der Virtualität über, spricht man von Augmented Virtuality. Sowohl Augmented Reality als auch Augmented Virtuality fallen nach Milgram und Kishino unter die Definition „Mixed Reality", welche alle Systeme umfasst die Realität und Virtualität miteinander vermischen. Zuletzt sei noch erwähnt, dass Augmented Reality nicht zwangsläufig auf die visuelle Ebene beschränkt ist. So existieren auch Anwendungsfälle in denen andere Sinnesreize virtuell erweitert werden. In dieser Arbeit beschränkt sich die Definition auf die visuelle Variante.

2.1.2 Technische Voraussetzungen

Nachdem Augmented Reality definiert wurde, stellt sich noch die Frage nach der Art und Weise der technischen Realisierung. Es existieren zahlreiche Ansätze um Augmented Reality zu erreichen. Das System hängt hierbei stark vom Anwendungsfall ab. Prinzipiell kann man jedoch ein AR-System grob in die folgenden Komponenten aufteilen[8]:

[8]Vgl. Dorner, *Virtual und Augmented Reality (VR/AR): Grundlagen und Methoden der Virtuellen und Augmentierten Realität (eXamen.press) (German Edition)*, S.242

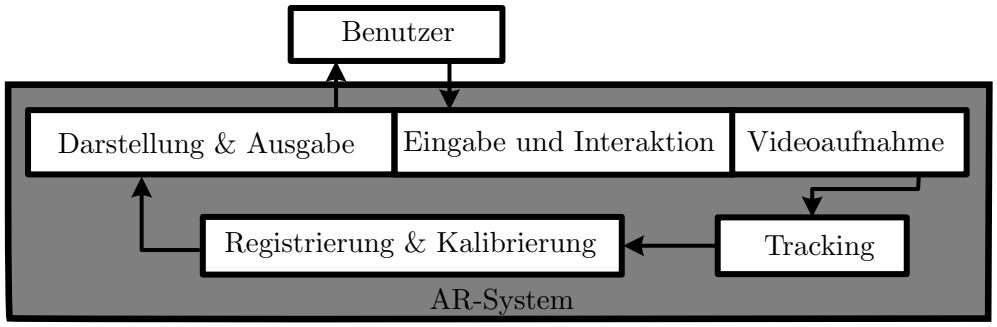

Abbildung 2: Die Grundbausteine eines AR-Systems. Eigene Darstellung in Anlehnung an Dorner[10].

Die wichtigste Komponente eines Augmented Reality Systems ist das Tracking. Es hat die Aufgabe die Position, Orientierung und die Lage des Betrachters zu ermitteln. Eine weitere wichtige Komponente ist die Registrierung und Kalibrierung. Nachdem die Lage des des Betrachters ermittelt wurde, müssen die virtuellen Inhalte generiert und auf Basis der Tracking-Informationen an der richtigen Stelle und in der richtigen Lage in das Blickfeld des Benutzers platziert werden. Schließlich muss das Bild auf einem Ausgabegerät gerendert werden. Optional existiert eine Eingabeschnittstelle, mit welcher der Benutzer interagieren kann. Auch eine Kamera zur Videoaufnahme ist nicht zwingend notwendig. Sie wird meistens benötigt um Video-See-Through zu ermöglichen (siehe 2.1.2.1) oder um bildbasierte Trackingverfahren einzusetzen. Im Folgenden werden die wichtigsten Komponenten eines AR-Systems genauer erläutert.

2.1.2.1 Darstellung & Ausgabe

Die Ausgabe eines Augmented Reality Systems charakterisiert die Art und Weise, in welcher dieses die Verschmelzung von Realität und Virtualität erreicht. Man unterscheidet die folgenden Arten der Ausgabe[11][12]:

1. **Optisches See-Through**
2. **Video See-Through**
3. **Räumlich**

Beim optischen See-Through Verfahren nimmt der Benutzer die reale Welt mit den eigenen Augen wahr. Ein optischer Kombinierer sorgt dafür,

[10] Vgl. Dorner, *Virtual und Augmented Reality (VR/AR): Grundlagen und Methoden der Virtuellen und Augmentierten Realität (eXamen.press) (German Edition)*, S.242
[11] Vgl. a. a. O., S.248
[12] Vgl. Bimber und Raskar, *Spatial Augmented Reality: Merging Real and Virtual Worlds: A Modern Approach to Augmented Reality*, S.6

dass die reale Welt durch virtuelle Inhalte überblendet werden[13]. Es existieren verschiedene technische Ansätze für optische Kombinierer. Oftmals werden solche mithilfe von semi-transparenten, reflektierenden Displays realisiert, die nur einen Teil der von vorne einfallenden Lichtstrahlen passieren lassen, um einen Teil des Lichtes aus dem Anzeigegerät heraus in das Auge reflektieren zu können[14].

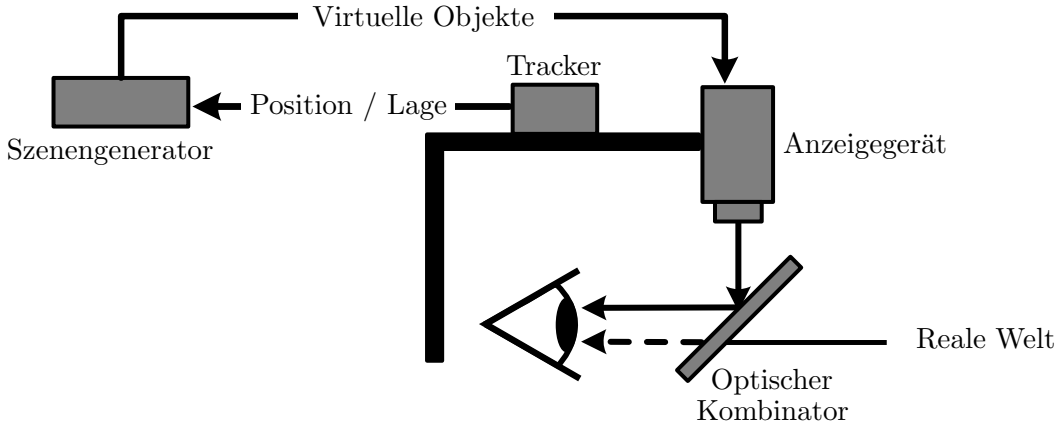

Abbildung 3: Optisches See-Through Verfahren. Eigene Darstellung in Anlehnung an Azuma [16].

Der Hauptvorteil dieser Methode liegt darin, dass der Benutzer immer die reale Welt mit eigenen Augen sieht und daher kein verzögertes Bild der Realität erhält. Dies geht allerdings mit dem Nachteil daher, dass das virtuelle Bild sehr kurze Antwortzeiten liefern muss um bei der Überlagerung keine Verzögerungen zu erzeugen. Aufgrund dieser Tatsache stellt diese Form der Ausgabe deutlich höhere Anforderungen an das Tracking, Rendering und die Verarbeitungszeit des Rechners.
Ein weiterer Nachteil bei der Nutzung von semi-transparenten Displays ist das transparente Bild, wodurch virtuelle Inhalte unecht wirken. Dies macht sich insbesondere bei besonders hellen und besonders dunklen Umgebungen bemerkbar. Entweder ist das natürliche Licht zu stark und die virtuellen Inhalte verblassen, oder es ist zu schwach und das Licht der Anzeigekomponente überstrahlt die reale Umgebung.
Ein Beispiel für die Realisierung von Optical-See-Through sind Head-Mounted-Displays. Diese werden direkt am Kopf getragen und das Bild wird in der Nähe des Auges projiziert. Ein anderes Beispiel sind Head-Up-Displays (HUD), die häufig in Flugzeugen oder neuerdings auch in Autos zu finden sind. Hierbei befindet sich der optische Kombinierer häufig weiter vom

[13] Vgl. Azuma, Presence: Teleoperators and Virtual Environments, Nr. 4, Bd. 6, 1997, S.455-464
[14] Vgl. a. a. O.
[16] A. a. O.

menschlichen Auge entfernt.

Beim Video-See-Through sieht der Benutzer immer nur ein aufgenommenes Echtzeit-Video der Realität[17]. Ein solches AR-System besteht zwingend aus einer Video-Kamera, die das aktuelle Live-Bild erfasst und einem Tracking-System, welches die Lage und Position ermittelt. Anhand der Trackingdaten werden die virtuellen Objekte korrekt berechnet und mithilfe eines Videokombinierers mit dem Videobild verschmolzen. Das finale Bild wird dann im Anzeigegerät ausgegeben.

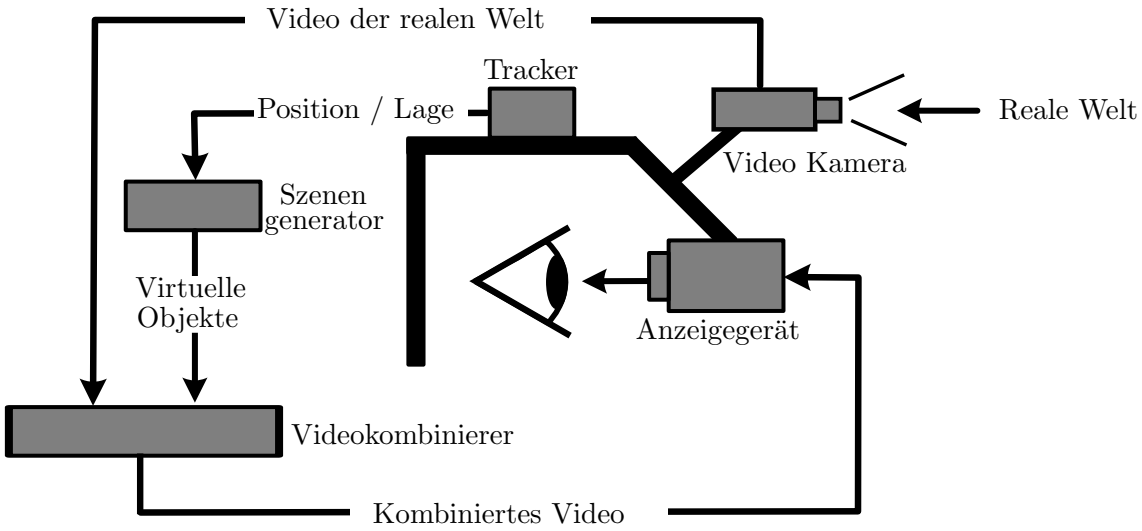

Abbildung 4: Video See-Through Verfahren. Eigene Darstellung in Anlehnung an Azuma [19].

Durch die Möglichkeit, das von der Video-Kamera aufgenommene Bild Pixel-genau modifizieren zu können[20], gestaltet sich die Verschmelzung von Virtualität und Realität deutlich einfacher. Die Überlagerung von virtuellen Inhalten wird nicht durch äußere Lichteinflüsse beeinflusst, so wie das beim optischen See-Through-Verfahren der Fall ist. Zudem kann eine durch Tracking- und CPU-Rechenzeit verursachte Bearbeitungsverzögerung mithilfe einer geringeren Bildrate kompensiert werden. Auf diese Weise kann erreicht werden, dass die virtuellen Inhalte sich zwangsläufig im Live-Bild korrekt einfügen. Dies hat jedoch zur Folge, dass der Benutzer das Gesamtbild verzögert erhält. Es muss daher stets ein Kompromiss zwischen niedriger Bildrate und Schwimmeffekten gefunden werden.
Es gibt verschiedene Möglichkeiten Video-See-Through zu realisieren. In der Praxis können hierfür HMD's, HUD's oder herkömmliche PCs oder PDA's

[17] Vgl. Azuma, Presence: Teleoperators and Virtual Environments, Nr. 4, Bd. 6, 1997, S.455-464
[19] A. a. O.
[20] Vgl. a. a. O., S.455-464

zum Einsatz kommen. Besonders häufig finden sich Video-See-Through Anwendungen aber auf Tablet-PCs oder Smartphones.

Räumliche oder umgebungsfixierte[21] Verfahren unterscheiden sich von den vorherigen vorgestellten Verfahren dadurch, dass die virtuellen Inhalte vom Betrachter losgelöst im freien Raum visualisiert werden. Mithilfe spezieller Projektionstechniken lassen sich auf beliebige Oberflächen virtuelle Inhalte projizieren. Der Vorteil solcher Systeme liegt in der Möglichkeit diese kollaborativ zu nutzen. Aufgrund der aufwendigen Apparatur sind sie für den mobilen Einsatz jedoch nicht geeignet.

2.1.2.2 Tracking

Das Tracking stellt die wichtigste Grundlage eines AR-Systems dar. Im Idealfall liefert es alle sechs Freiheitsgrade einer Pose, also drei Koordinaten für die Position und drei Winkel für die Lage. Durch diese kann ein Bezug zum realen dreidimensionalen Raum hergestellt und virtuelle Inhalte Kontext-abhängig registriert und visualisiert werden.

Die Wahl eines oder mehrerer geeigneter Trackingverfahren ist eine Designentscheidung, die vom Anwendungsfall abhängt. Ortsgebundene AR-Anwendungen werden in der Regel mit laufzeitbasierten in Kombination mit sensorbasierten Systemen realisiert, um ortsabhängig Inhalte in der Umgebung zu visualisieren. Objektabhängige AR-Anwendungen werden meist mit optischen Verfahren realisiert.

Man unterscheidet generell zwischen den folgenden Tracking-Verfahren:

1. **Mechanisch**
2. **Optisch (markerbasiert oder markerlos)**
3. **Laufzeitbasiert**
4. **Sensorbasiert**

Das mechanische Tracking wurde bereits 1965 von Ivan Sutherland eingesetzt[22]. In der Regel werden hierfür aus mehreren Gelenken und Metallstangen bestehende mechanische Arme eingesetzt. In jedem Gelenk wird der Rotationswinkel kontinuierlich gemessen. Im Zusammenhang mit den Längeninformationen der Metallstäbe wird daraus die relative Position der Gelenke zur Basis ermittelt[23].

[21] Vgl. Tönnis, *Augmented Reality*, S.25
[22] Vgl. Sutherland, A Head-mounted Three Dimensional Display
[23] Vgl. Bormann, *Virtuelle Realität: Genese und Evaluation*, S.57-71

Der Nachteil solcher Konstruktionen ist die eingeschränkte Bewegungsfreiheit und Mobilität. Neben solchen feststehenden Konstruktionen, werden daher auch tragbare Systeme wie bspw. Exoskellete[24] eingesetzt.

Unter dem optischen Tracking versteht man das Ermitteln der Pose mittels Videobildanalyse. Eine Videokamera erfasst dabei das Bild entweder relativ zum Objekt (inside-out) oder vom Objekt losgelöst (outside-in)[25]. Ein Beispiel für ein Inside-Out-System wäre ein Video-See-Through HMD. Ein Outside-In-System könnte beispielsweise eine Fußballübertragung mit AR-Elementen sein.
Beim optischen Tracking werden prinzipiell zwischen markerbasiertem und markerlosem Tracking[26][27] unterschieden.

Das markerbasierte Tracking ist ein optisches Tracking-Verfahren welches die Pose relativ zur Kamera ermittelt. Um die Pose zu bestimmen, werden sogenannte Marker eingesetzt. Marker sind Objekte, die durch ihre Art und Form leicht durch eine Kamera identifiziert werden können[28]. Solche Marker weisen daher eindeutige Merkmale wie z. B. Helligkeit, Reflektioncharakteristiken, Farbe oder Kontrast auf, um diese im Videobild von anderen Objekten unterscheiden zu können. Dabei unterscheidet man wiederum zwischen aktiven und passiven Markern. Aktive Marker erzeugen selbst künstliches Licht um diese im Videostrom herausfiltern zu können. In der Regel wird hierfür Infrarotlicht eingesetzt, da dieses vom menschlichen Auge nicht sichtbar ist. Passive Marker bestehen entweder aus retroreflektiven Materialien, sodass kein zusätzliches Licht benötigt wird, oder aus bestimmten Mustern, die mittels Bildanalyse identifiziert werden[29]. Letztere sind besonders kosteneffizient zu realisieren.
Markerbasierte Trackingsysteme setzen zwingend Marker voraus, die vorher an den jeweiligen Stellen platziert werden müssen. Da dies nicht immer wünschenswert bzw. realisierbar ist, müssen häufig auch markerlose Verfahren zum Einsatz kommen. Eines dieser Verfahren ist das Feature-Tracking[30]. Solche Trackingsysteme verwenden natürliche Merkmale, sogenannte „Features“, wie Kanten, Ecken um die Pose zu bestimmen[31]. An-

[24] Vgl. o. V., Gypsy Motion capture system technology explained
[25] Vgl. Dorner, *Virtual und Augmented Reality (VR/AR): Grundlagen und Methoden der Virtuellen und Augmentierten Realität (eXamen.press) (German Edition)*, S.107-109
[26] Vgl. Tönnis, *Augmented Reality*, S.45-52
[27] Vgl. Dorner, *Virtual und Augmented Reality (VR/AR): Grundlagen und Methoden der Virtuellen und Augmentierten Realität (eXamen.press) (German Edition)*, S.104-107
[28] Vgl. Mehler-Bicher, Reiß und Steiger, *Augmented Reality: Theorie und Praxis*, S.29
[29] Vgl. Dorner, *Virtual und Augmented Reality (VR/AR): Grundlagen und Methoden der Virtuellen und Augmentierten Realität (eXamen.press) (German Edition)*, S.104
[30] Vgl. Furht, *Handbook of Augmented Reality*, S.255
[31] Vgl. Dorner, *Virtual und Augmented Reality (VR/AR): Grundlagen und Methoden der Virtuellen und Augmentierten Realität (eXamen.press) (German Edition)*, S.107

hand von Eck-, Kanten-, oder Blobdetektoren werden Merkmale im Bild ausfindig gemacht. Canny-Detektoren werden eingesetzt, um die Kanten aus dem Bild zu extrahieren. Featuredetektoren wie FAST, um Ecken zu markieren, oder SIFT, um sogenannte Keypoints herausfiltern können[32]. Die Merkmalregionen werden anschließend durch aussagekräftige, eindeutige und invariante Eigenschaften beschrieben (sogenannte Deskriptoren), die unabhängig von ihrer Position, Rotation oder Skalierung miteinander vergleichbar sind (Feature Description)[33]. Auf diese Weise können die Deskriptoren in jedem weiteren Bild im Videostrom verfolgt werden (Feature Matching).
Ein weiteres Beispiel für ein markerloses Verfahren ist die Analyse durch Synthese. Hierbei wird ein reales Objekt nahezu identisch als virtuelles dreidimensionales Objekt nachgebildet. Durch die Möglichkeit, das virtuelle Modell beliebig rotieren zu können, und das daraus gerenderte Bild mit dem realen Objekt vergleichen zu können, ist ein solches System in der Lage die Pose des Objektes zu ermitteln[34].

Neben optischen, markerbasierten Tracking ist das laufzeitbasierte Tracking das am häufigsten verwendete Tracking-System. Darunter versteht man Systeme, die die gemessene Zeit eines Signals von einer Sendequelle zum Empfänger nutzen um die Position eines Objektes zu bestimmen[35]. Das wohl bekannteste laufzeitbasierte Tracking-System ist das Global Positioning System (GPS) . Dabei senden Satelliten ein Signal mit einem exakten Zeitstempel an einen Empfänger aus. Anhand eines Zeitstempels zur Empfangszeit kann der Empfänger die Übertragungszeit der Signale und somit die Entfernung des Satelliten zum Empfänger berechnen. Mit den entsprechenden Entfernungen zu allen Satelliten erhält der Empfänger Radiuskurven, dessen Schnittpunkt die Position darstellt[36]. Ein Beispiel für ein laufzeitbasiertes System innerhalb von Gebäuden ist das Wireless Local Area Network. Auch hier kann die Position anhand der Signallaufzeit von AccessPoints ermittelt werden. Zudem können Ultraschall- oder Infrarotsensoren genutzt werden, um die Entfernung zwischen Sender und Empfänger zu berechnen.

Sensorbasierte Trackingsysteme ermitteln die Pose eine Objektes mithilfe von spezieller Sensorik. Hierunter fällt die Inertialsensorik, also Gyroskope oder Beschleunigungssensoren, die die Drehraten und Beschleunigung mes-

[32] Vgl. Furht, *Handbook of Augmented Reality*, S.257-258
[33] Vgl. o. V., CS-9645 Introduction to Computer Vision Techniques Winter 2014, S.1-2
[34] Vgl. Schumann, Achilles und Müller, Analysis by Synthesis Techniques for Markerless Tracking, S.4
[35] Vgl. Tönnis, *Augmented Reality*, S.54
[36] Vgl. a. a. O., S.55

sen, wie auch Magnetometer, die in der Lage sind magnetische Ströme zu erfassen.
Intertialsensoren machen vom Massenträgheitsgesetz Gebrauch, um die rotatorischen und translatorischen Bewegungsänderungen eines Objektes zu ermitteln. In der Regel werden mehrere Inertialsensoren zu einer Inertial Measurement Unit (IMU) zusammengefasst. Diese bestehen meist aus drei orthogonal ausgerichteten Beschleunigungs- und Gyroskopsensoren, die die lineare Beschleunigung und die Drehrate in allen drei Achsen (x,y,z) messen[37].
Magnetfeldsensoren, sogenannte Magnetometer, richten sich entweder dem Magnetfeld der Erde oder nach einem künstlich erzeugten Magnetfeld aus, woraus ein Richtungsvektor ermittelt werden kann[38]. In der Regel werden jedoch künstliche Magnetfelder erzeugt, da das Erdmagnetfeld durch störende Faktoren wie z.B. die Sonne keinen extakten Verlauf hat.[39]. Inertialsensoren liefern immer nur eine relative Messung zum Ausgangspunkt. Durch die zweifache Integration der Beschleunigung akkumulieren sich die Fehler der relativen Messung quadratisch auf, sodass schnell ein sogenannter Drift entsteht. Dieser lässt sich durch einen magnetischen Referenzpunkt kompensieren. Magnetometer werden daher oftmals in Kombination mit Inertialsensorik eingesetzt.

2.1.2.3 Kalibrierung & Registrierung

Eine weitere wichtige Komponente eines AR-Systems ist die Kalibrierung & Registrierung. Unter Registrierung versteht man das korrekte und möglichst exakte Einfügen von virtuellen Inhalten in das reale Bild. Hierfür ist es notwendig die Ausgabedaten der Tracking-Systeme genau aufeinander abzustimmen. Dies stellt eine besondere Herausforderung dar, da jedes Tracking System einem eigenen Koordinatensystem unterliegt. Es müssen also geeignete Parameter gefunden werden, um die Posedaten der Tracking-Systeme in das Koordinatensystem des Renderingsystems zu überführen[40].

Bei Optical-See Through Displays kann es zusätzlich notwendig sein, die Koordinaten der Augen mit denen der Kamera abzustimmen, da sich der Koordinatenursprung des Auges bspw. durch Bewegungen des Nutzers verschieben könnte. Es müssen also rechtzeitig Divergenzen identifiziert und korrigiert werden. So können beispielsweise auch radiale oder tangentiale Verzerrungen durch die Kameralinse entstehen, die durch entsprechende Transformationsparameter kompensiert werden müssen. In der Literatur

[37] Vgl. Tönnis, *Augmented Reality*, S.52-53
[38] Vgl. a. a. O., S.54
[39] Vgl. a. a. O., S.52-53
[40] Vgl. Suthau, Augmented Reality - Positionsgenaue Einblendung räumlicher Informationen in einem See Through Head Mounted Display für die Medizin am Beispiel der Leberchirurgie, S.64

wird dieser Prozess gelegentlich auch als „Kalibrierung“ bezeichnet. Generell geht die Definition der Kalibrierung und Registrierung jedoch fließend über.

Bei optischen Trackingsystemen unterscheidet man zwischen zwei Formen von Faktoren: intrinsische (bzw. geometrische) und extrinsische[41] Parameter. Bei intrinsischen Faktoren wird der Zusammenhang zwischen 3D- und 2D-Bildkoordinaten wiederhergestellt. Hierunter fallen zum Beispiel Faktoren, die Verzeichnungen bei der Kameralinsenverzerrung[42] korrigieren oder solche, die die Brennweite der Kamera und die Lage des Bildzentrums ausgleichen[43]. Extrinsische Parameter wiederum, stellen den Zusammenhang zwischen Welt- und dem Kamerakoordinatensystem her[44].

2.1.2.4 Eingabe und Interaktion

Oftmals setzt ein Augmented Reality System auch eine Benutzerschnittstelle voraus, um eine Benutzerinteraktion zu ermöglichen. Einfache Eingabegeräte wie die Tastatur oder Maus eignen sich für AR-Systeme in den meisten Anwendungsfällen nur bedingt, da AR Systeme oftmals Anforderungen an Mobilität und dreidimensionaler Interaktivität stellen. Es wurden daher neue Ansätze entwickelt, um eine Benutzer-Interaktion zu realisieren.
Die meist verbreitete Form der AR-Interaktion ist das Tangible User Interface[45]. Wie der Name schon vermuten lässt, handelt es sich hierbei um eine „anfassbare“ Form der Interaktion. Oftmals wird diese mithilfe von optischen Trackingverfahren realisiert, wodurch reale Objekte als Anker fungieren, um virtuelle Objekte beliebig manipulieren zu können. Sollen nur einfache Operationen, wie Verschiebung und Rotation möglich sein, sind hierfür einfache Marker als Ankerpunkt bereits ausreichend. So kann aus mehreren Markern bereits ein vollständiges Graphical User Interface(GUI) erzeugt werden. Bei komplexeren Operationen eignen sich z. B. mit aktiven oder passiven Markern versehene Datenhandschuhe, mit welchem durch Hand- bzw. Fingerbewegungen virtuelle Inhalte selektiert und manipuliert werden können. Auch eine Gestensteuerung ist denkbar.
Bei Video-See-Through Systemen, die auf Smartphones oder Tablets zum Einsatz kommen, bieten sich Touch-Gesten zur Interaktion an. Feststehende Systeme verwenden oft 3D-Mäuse oder mechanische Konstruktionen, die bereits in 2.1.2.2 beschrieben wurden.

[41] Vgl. o. V., 3DReco Homepage
[42] Vgl. o. V., Vektorgrafik, Verschlusszeit, Verzeichnung, Vignettierung, Vollautomatischer Weißabgleich bei Digitalkameras, Vollformatsensor
[43] Vgl. o. V., 3DReco Homepage
[44] Vgl. a. a. O.
[45] Vgl. Dorner, *Virtual und Augmented Reality (VR/AR): Grundlagen und Methoden der Virtuellen und Augmentierten Realität (eXamen.press) (German Edition)*, S.285 ff.

Grundsätzlich kann neben der klassischen Handsteuerung auch der gesamte Körper zur Interaktion genutzt werden. So könnte z. B. ein Exoskelett durch Erkennung von bestimmten Bewegungsmuster fest definierte Events auslösen, sodass das AR-System kontextbezogene Informationen generieren kann. Zudem werden auch Eye-Tracking-Systeme eingesetzt, um die Blickrichtung des Benutzers zu erfassen. Auf diese Weise kann ein längerer verharrter Blick auf ein Objekt als Selektion interpretiert werden. Da jedoch Eye-Tracking Systeme immer höhere Anforderungen an die Kalibrierung stellen, kann stattdessen die Orientierung des Kopfes bzw. HMD's oder des Handheld Gerätes genutzt werden um eine Interaktion zu ermöglichen[46]. Beispielsweise könnte durch das Verändern Kopfrichtung das Objekt mittig im Bild anvisiert und durch längeres Verharren selektiert werden.
Zuletzt kann die menschliche Sprache als Eingabemedium dienen. Da die automatische Spracherkennung heute eine hohe Genauigkeit erreicht stellt sie eine geeignete Alternative dar. Auf diese Weise können Funktionen freihändig ausgeführt werden. Solche Systeme gehen jedoch mit dem Nachteil daher, dass die Qualität der Benutzereingaben von Umgebungsgeräuschen beeinträchtigt werden.

2.2 Ausprägungen der AR auf mobilen Geräten

Wie bereits im letzten Kapitel dargelegt, gibt es unzählige Methoden Augmented Reality zu realisieren. Es wurden hier jahrelang vielversprechende Ansätze vorgestellt, die sich jedoch im Alltag aufgrund der aufwendigen, schweren und auffälligen Apparatur nur bedingt eigneten. Bedingt durch den rasanten Fortschritt der Rechnertechnik und der damit einhergehende Aufstieg des mobilen Marktes, konnte sich Augmented Reality im mobilen Umfeld erstmals als alltagstaugliche Technologie etablieren.
Generell, bringen heute Handhelds, also Smartphones und Tablets neben einer Kamera standardmäßig viele Komponenten mit, um ein Video-See-Through Augmented Reality System zu realisieren. Als Beispiele seien hier die Inertialsensorik, wie Gyroskope, Accelerometer, als auch die Magnetfeldsensoren oder GPS zu nennen.
Dennoch wurden bis heute nicht alle Tracking-Technologien unterstützt, genauso ist die Leistung der Smartphone-Hardware noch nicht in allen Fällen ausreichend. Diese Entwicklung hatte zur Folge, dass AR-Systeme im mobilen Bereich auf die technischen Gegebenheiten der Geräte beschränkt waren und sich daher an solche entsprechend anpassen mussten.
Aus diesen Rahmenbedingungen heraus haben sich im mobilen Umfeld ge-

[46] Vgl. Dorner, *Virtual und Augmented Reality (VR/AR): Grundlagen und Methoden der Virtuellen und Augmentierten Realität (eXamen.press) (German Edition)*, S.285 ff.

nerell zwei Ausprägungen der AR entwickelt, die im Folgenden vorgestellt werden sollen.

2.2.1 Objektbasierte AR

Die objektbasierte Form der erweiterten Realität wird durch visuelle Trackingverfahren erreicht. Grundlegend unterscheidet sich das objektorientierte Tracking vom ortsbasierten Tracking darin, dass die Erweiterung der visuellen Realitätswahrnehung nicht durch Informationen über dem räumlichen Kontext erfolgt, sondern reale Objekte als Anhaltspunkt und Posebestimmung genutzt werden. Hierfür kommen die markerbasierten oder markerlosen Verfahren zum Einsatz, die zuvor in 2.1.2.2 beschrieben wurden.
Im Bereich der Printmedien haben sich besonders Marker zur Erweiterung der Realität durch multimediale Inhalte bewährt. Diese Unterform wird auch als „Augmented Printing “[47] bezeichnet. Doch auch ohne Marker können virtuelle Objekte bereits korrekt registriert werden. Sicherlich kann hier als Beispiel aus der Praxis eine App zur Einrichtung von Möbeln in Räumen herangezogen werden.

2.2.2 Ortsbasierte AR

Bei der ortsbasierten AR werden virtuelle Inhalte über den räumlichen Kontext bezogen. Der räumliche Bezug erfolgt dabei losgelöst von physikalischen Objekten, über die Position und Pose des mobilen Geräts. Bei dieser Form der AR liegt der Schwerpunkt daher vor allem auf Verfahren zur Positionsbestimmung. Die Funktionsweise eines solchen AR-Systems ist wie folgt: Zunächst werden an verschiedenen Positionen bzw. Punkte in einem räumlichen Koordinatensystem Informationen hinterlegt. Anhand eines standortbezogenen Dienstes wird die Position des Benutzers innerhalb des Koordinatensystems ermittelt. Mithilfe der Position und der hinterlegten Informationen im Koordinatensystem, kann der Peilwinkel ermittelt werden. Zudem kann vom Standort aus, mithilfe der in Handhelds vorhandenen Sensorik, die Blickrichtung des Handheld-Gerätes erfasst werden. Alle Informationen werden schließlich zusammengeführt, um die Position der virtuellen Objekte im Bild berechnen zu können.
Unter den Beispielen für ortsbasierte AR finden sich zahlreiche Apps für das Auffinden von sogenannten Points of Interests(POI's), wie Restaurants, Taxen oder Informationen zu Sehenswürdigkeiten, wie auch AR-Spiele, darunter das von Google entwickelte Spiel Ingress.

[47] Vgl. Rose M., Augmented Reality und Printmedien

2.3 Indoor-Navigation

Die Navigation bezeichnet die optimale Wegfindung von einer gegebenen Position zu einem bestimmten Zielort. Diese setzt sich aus den folgenden drei Teilbereichen zusammen:

1. **Das Bestimmen der Position innerhalb eines zwei- bzw. dreidimensionalen Koordinatensystems**
2. **Das Berechnen eines optimalen Pfades zum gewünschten Ziel**
3. **Das Führen des Benutzers zum gewünschten Ziel**

Im allgemeinen Sinne wird die Navigation auch als das Zurechtfinden innerhalb eines topographischen Raums verstanden. Darunter fallen auch bspw. Beschilderungen, die einem Benutzer helfen, sich innerhalb eines Einkaufzentrums zurechtzufinden und dazu beitragen, dass dieser sein gewünschtes Ziel erreicht. Die ersten beiden Teilbereiche müssen also nicht zwangsläufig erfüllt sein, um Navigation zu ermöglichen. Diese Arbeit betrachtet auch diesen Aspekt. Die Indoor Navigation beschränkt sich dabei ausschließlich auf die Navigation innerhalb von geschlossenen Räumen. Dies stellt eine besondere Herausforderung dar, da auf engerem Raum deutlich höhere Anforderungen an die Genauigkeit gestellt werden. Während es im Outdoor-Bereich bei der Suche nach einem Gebäude auf eine Abweichung von 5 Metern nicht ankommt, würde man im Indoor Bereich mit einer derartigen Abweichung u. U. bereits mehrere Räume weiter lokalisiert werden. Selbstverständlich hängt die Genauigkeit auch von dem jeweiligen Anwendungsfall ab.

2.3.1 Standortbestimmung

Zum jetzigen Zeitpunkt existiert noch kein einheitlicher Standard zur Indoor Positionierung. Im Außenbereich ist GPS in Verbindung mit GSM und WLAN der de-facto Standard. Im Indoor-Bereich eignet sich GPS allerdings nicht. Zum einen liegt die Genauigkeit eines GPS-Systems bei etwa 5-20 Metern und reicht daher für den Indoor Einsatz nicht aus. Zum anderen wird das GPS-Signal der Satelliten innerhalb von Gebäuden je nach Baumaterial des Gebäudes gedämpft. Um also eine präzise Lokalisierung innerhalb von Gebäuden zu erreichen, müssen alternative Verfahren zum Einsatz kommen. Die Abbildung 5 gibt eine Übersicht über mögliche Verfahren.

Die meisten in der Literatur existierenden Verfahren basieren auf Funk. Dies ist sicherlich auf die Tatsache zurückzuführen, dass Funknetze in den

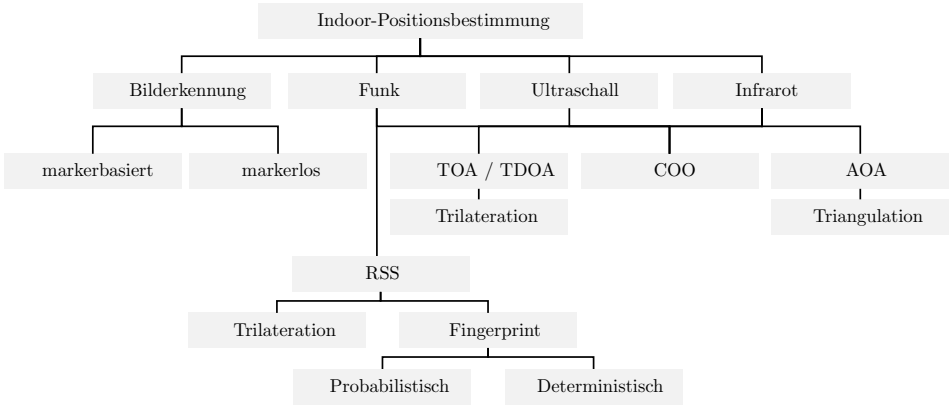

Abbildung 5: Übersicht über mögliche Verfahren zur Indoor-Positionsbestimmung. Eigene Darstellung.

meisten Gebäuden bereits zur Verfügung stehen. Infrarot basierte Verfahren haben den Nachteil, dass sie sehr Lichtanfällig sind und dass der Sender immer Sichtkontakt zum Empfänger haben muss. Ultraschall-basierte Verfahren sind wiederum von Umgebungsgeräuschen abhängig. Zwar existieren Ansätze mit Schall oder Infrarot, die sich jedoch aufgrund der benötigten Infrastruktur in der Praxis nicht durchgesetzt haben. Bei Funk-basierten Verfahren unterscheidet man oftmals zwischen Infrastruktur-basierten oder Endgeräte-basierten Verfahren, je nachdem an welchem Endpunkt die Verarbeitung stattfindet.

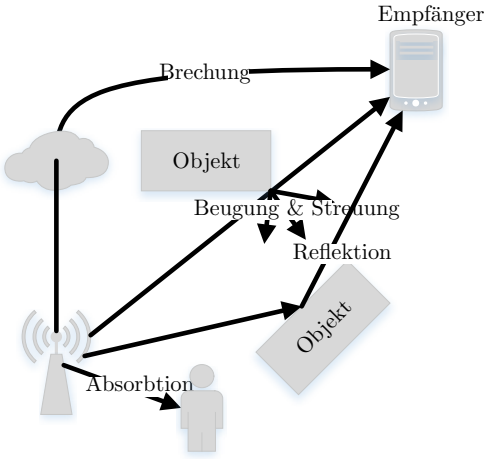

Abbildung 6: Mehrwegeausbreitungen bei Funk-basierten Verfahren

Die größte Herausforderung an funkbasierten Verfahren sind allerdings die Pfadverluste[48], die Funksignale auf dem Weg vom Sender zum Empfänger erleiden. Hier wäre zunächst der Effekt der Absorption zu nennen. Dabei

[48] Vgl. Popovic, Fachhochschule Wedel – Informatikseminar WS2004 2004, S.1 ff.

werden Signale beim Versuch Objekte zu durchqueren von bestimmten Materialien vollständig absorbiert, wie bspw. Regentropfen oder Menschen, die zum Großteil aus Wasser bestehen. Ein weiteres Problem ist die Reflexion. Objekte mit reflektiven Eigenschaften, wie etwa metallischen Gegenstände führen dazu, dass sich Signale von Objekt zu Objekt hangeln, bis sie den Empfänger erreichen, wobei die Stärke Signals immer weiter abnimmt. Ferner entstehen durch Objekte Beugungsverluste, bei denen Signale von Objekten abgelenkt werden, oder Streuverluste, bei denen ein Signal in kleinere, schwächere Signale aufteilt wird. In der Praxis treten alle Effekte in Kombination auf und verursachen unzählig viele Ausbreitungspfade der Signale. Diese Mehrfachausbreitung der Signale erschwert daher die exakte Positionsbestimmung maßgeblich. Insbesondere innerhalb von Gebäuden, sind solche Effekte aufgrund der engen Dichte der Objekte wie Wände und Raumgegenstände stärker ausgeprägt.

Im Folgenden sollen die verbreitetsten Verfahren im Detail erläutert, und dazu bereits existierende Arbeiten vorgestellt werden.

2.3.1.1 Time of Arrival / Time Difference of Arrival

Time of Arrival (zu deutsch "Ankunftszeit„ , kurz TOA) bezeichnet Verfahren, die die Entfernung zwischen einer Sendestation und einem Empfangsgerät auf Basis der Signallaufzeit ermitteln. TOA nutzt dabei die absolute Ankunftszeit eines Signals um die Entfernung ermitteln zu können. In der Praxis wird das Signal jedoch durch äußere Faktoren wie Reflexion oder Dämpfung beeinflusst. Wichtig bei diesem Verfahren ist eine exakte Zeitmessung von $\leq 1ns$, da ansonsten die Genauigkeit um mehrere 100 Meter abweicht. Ferner müssen Empfänger und Sender immer zeitlich synchronisiert werden. TOA wird in der Regel bei GPS verwendet, da im Freien die Ausbreitungsgeschwindigkeit relativ konstant ist. Im Indoor-Bereich existieren bereits Ansätze TOA mit WLAN oder Bluetooth zu realisieren, allerdings scheitern sie meist aus Kostengründen. Izquierdo et al.[49] haben gezeigt, dass eine Indoor-Positionierung mittels WLAN und TOA möglich ist. Dabei wurde die Round Trip Time von RTS-Frames als Signallaufzeit definiert. Ihr System erreichte eine Genauigkeit von etwa zwei Metern. Allerdings mussten hierfür Anpassungen an den Access Points vorgenommen werden. Auch mit Bluetooth konnte über TDOA eine Genauigkeit von etwa 1 Meter nachgewiesen werden[50]. Ein in der Literatur existierendes Beispiel für den Einsatz von TDOA mit Ultraschall ist das System Active Bat[51].

[49] Vgl. Izquierdo F. und E., Performance evaluation of a TOA-based trilateration method to locate terminals in WLAN, S.1 ff.
[50] Vgl. Fischer G., Bluetooth Indoor Localization System, S.1 ff.
[51] Vgl. o. V., Active Bat | Indoor-Ortung.de

Analog zu Time of Arrival existiert das Time Difference of Arrival (kurz TDOA). Dieses unterscheidet sich von TOA dadurch, dass nicht die absolute Ankunftszeit sondern die Zeitdifferenz zwischen dem Absenden des Signals bis zum Empfang gemessen wird. Dies wird mithilfe eines Zeitstempels realisiert, der die Sendezeit im Signal codiert. Grundsätzlich gelten hier die selben Voraussetzungen wie bei TOA, die eine Synchronisation und genaue Zeitmessung notwendig macht.
Die Entfernung zu einer einzigen Basisstation reicht für die Positionsbestimmung allerdings nicht aus, da dadurch lediglich die ungefähre Position im Umkreis der Station ermittelt wird. Dies lässt sich bildlich (siehe Abb. 7) durch einen Kreis visualisieren, dessen Radius die Entfernung vom Gerät zur Basisstation darstellt. Wenn nun die Entfernung zu mehreren Basisstationen bekannt ist entstehen bildlich auch mehrere Kreise. Schneiden sich mindestens drei dieser Kreise, dann stellt der Schnittpunkt die theoretische Position des Empfangsgerätes dar.

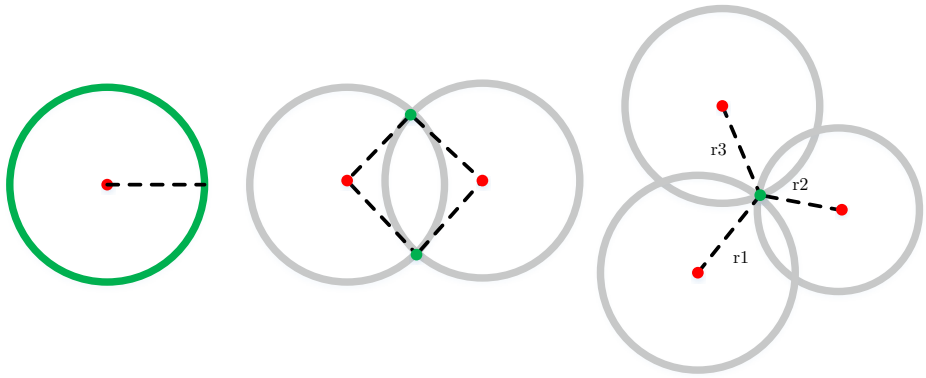

Abbildung 7: Lateration und Multilateration. Eigene Darstellung.

Diese Methode wird als Lateration bzw. Multilateration bezeichnet. Ausgangspunkt für die Berechnung ist die Kreisgleichung

$$r^2 = (x - m_1)^2 + (y - m_2)^2 \tag{1}$$

mit dem Radius r, welcher den Abstand zwischen Basisstation und Empfangsgerät beschreibt und dem Punkt $M = (m_1|m_2)$, der die Position der Basisstation darstellt. Die Position der Basisstation und die Distanz muss also bekannt sein. Die unbekannte Position des Empfangsgeräts $P = (x|y)$ auf dem Kreisrand ermittelt man durch Berechnung der Schnittpunkte aller Kreise in untereinander. Pro Paar wird ein Gleichungssystem aufgestellt, somit entstehen drei Gleichungssysteme mit jeweils zwei Kreisgleichungen. Daraus resultieren maximal sechs Schnittpunkte, also jeweils zwei pro Kreis. Da nur genau ein Schnittpunkt pro Kreis als Position in Frage kommen

kann, muss der andere Punkt mithilfe einer speziellen Heuristik ausgeschlossen werden. Beispielsweise werden die Schnittpunkte hinzugezogen, die sich vom Abstand untereinander am nächsten liegen. Der Schwerpunkt zwischen den drei ermittelten Schnittpunkten stellt schließlich die Position des Endgeräts dar. Da sich ein Funksignal i.d.R. kugelförmig ausbreitet, ist es denkbar eine zusätzliche Dimension aufzunehmen. Gerade im Indoor-Bereich, der sich oftmals über mehrere Etagen erstreckt, kann eine dritte Dimension sinnvoll sein, um die Genauigkeit zu verbessern. Die Standortbestimmung über die Lateration erreicht in der Regel eine Genauigkeit von 15 Metern. Die Ungenauigkeit ergibt sich vor allem durch die oben erwähnten Störfaktoren, die sich durch Dämpfungs- bzw. Reflektionseigenschaften der jeweiligen Objekte in der Umgebung ergeben.

2.3.1.2 Angle of Arrival

Beim Angle of Arrival (AOA wird der Eingangswinkel eines Signals genutzt um die Position eines Geräts zu ermitteln. Dabei müssen zwangsläufig zwei Basisstationen mit ihren Entfernungen zueinander bekannt sein. Solche Basisstationen besitzen spezielle Antennen-Arrays, bestehend aus mehreren unterschiedlich ausgerichteten Einzelantennen, die zu einem gleichen Zeitpunkt den Phasenunterschied zwischen den Signalen messen. Anhand der Phasenlage der Signale zueinander kann daraus der Eingangswinkel ermittelt werden. Das Angle of Arrival-Verfahren wird vor allem im GSM-Bereich genutzt, da hierfür spezielle Funkantennen mit Antennen-Arrays ausgestattet sind. Ein Praxisbeispiel von AOA ist Ubisense[52]. Es basiert auf Ultrabreitband-Technologie (UWB) und verwendet aktive Tags, die der Benutzer mit sich trägt. Diese senden in regelmäßigen Abständen UWB-Impulse aus. Spezielle Sensoren empfangen die Signale und können anhand des Eingangswinkel die Position des Benutzers ermitteln.
Ist der Eingangswinkel α einer Basisstation A, Eingangswinkel β der Basisstation B und die Entfernung $\overline{AB}$ der Basisstationen untereinander bekannt, so lässt sich daraus über Triangulation die Position bestimmen. Die Triangulation bedient sich dabei des einfachen Sinussatzes im Dreieck, um schließlich die Position C zu bestimmen:

$$\frac{A}{sin(\alpha)} = \frac{B}{sin(\beta)} = \frac{C}{sin(\gamma)} \tag{2}$$

2.3.1.3 Cell of Origin

Das Cell of Origin (kurz COO) Verfahren ermittelt die Position eines Gerätes anhand der angemeldeten Basisstation. Hierbei wird davon ausgegangen, dass jede Basisstation eine eindeutige ID und Position besitzt und ein

[52] Vgl. K. et al., An Evaluation of Indoor Location Determination Technologies

Gerät ist immer nur an genau einer Basisstation angemeldet ist. Ist zudem noch die Zellgröße des Funksignals bekannt, lässt sich auf diese Weise die ungefähre Position des Gerätes ableiten. Das Verfahren lässt sich mit GSM, wie auch mit Bluetooth und WLAN realisieren setzt jedoch voraus, dass eine flächendeckende Zellenstruktur aus mehreren Basisstationen besteht, um einen nahtlosen Übergang von einer Zelle in die andere zu ermöglichen, da ohne Anmeldung an einer Zelle folglich keine Positionsbestimmung möglich ist. Die Genauigkeit des COO-Verfahrens hängt stark von der Zellgröße und Zelldichte ab. Prinzipiell lässt sich COO mit so gut wie allen Funktechnologien realisieren, sofern das System eine ausreichende Zelldichte aufweist.

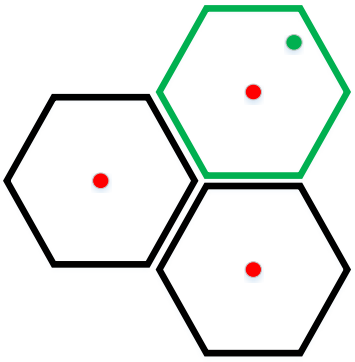

Abbildung 8: Cell of Origin. Eigene Darstellung.

Ein Beispiel aus der Praxis ist das Infrarot-basierte System ActiveBadge, das in den 90er Jahren in den AT&T Laboratories in Cambridge entwickelt wurde[53]. Es basiert auf einen Badge, den der Benutzer bei sich trägt und alle 15 Sekunden einen Infrarotimpuls an umliegende Sensoren verschickt. Die Daten werden Server-seitig ausgewertet. Das System erreicht allerdings nur eine Genauigkeit, die sich auf Räume beschränkt. Ein Beispiel für ein Bluetooth-basiertes Verfahren ist das System TOPAZ von der Firma Tadlys[54]. Der Benutzer trägt hierbei einen Tag bei sich, der im Umkreis von stationären Bluetooth AccessPoints lokalisiert wird. Die Genauigkeit wird zudem durch die hybride Nutzung mit Infrarot verbessert.

2.3.1.4 RSSI

RSSI steht für Received Signal Strength Indicator und misst die Empfangsfeldstärke die ein Gerät von einer Funk-basierten Infrastruktur zu einer bestimmten Zeit und einem bestimmten Ort empfängt. Der Wert hat keine vorgegebene Einheit, wird jedoch meist mit einem Geräteabhängigem

[53] Vgl. o. V., Active Badge | Indoor-Ortung.de
[54] Vgl. Dr. Weissman, Indoor Location

Skalierungsfaktor in dBm angegeben. RSSI-basierte Lokalisierungsverfahren nutzen den RSSI als Anhaltspunkt, um die Entfernung vom Sendestation zum Gerät ableiten zu können. In der Regel müssen hierfür die Sendeleistungen von mehreren Access Points an einer bestimmten Position gesammelt und ausgewertet werden. Um schließlich die Signalstärke für die Indoor-Positionsbestimmung verarbeiten zu können, existieren wieder unterschiedliche Verfahren. Zum einen die oben beschriebene Lateration und zum andern das sogenannte Tabellen-basierte Verfahren.
Geht man davon aus, dass sich durch Entfernen von der Basisstation die Sendeleistung abnimmt und durch Nähern entsprechend zunimmt, so ist es möglich hieraus ein theoretisches Distanzmaß zu bilden. Die Distanz von der Sendestation zum Empfangsgerät stellt dann gleichzeitig den Radius eines Kreises dar, entlang welchem man in der Theorie immer die gleiche Signalstärke empfängt und daher die Distanz zur Sendestation konstant bleibt. In der Praxis wird jedoch die Sendeleistung immer durch verschiedene Störfaktoren wie Streuung, Brechung oder Dämpfung beeinflusst. Die Schwierigkeit besteht daher darin ein Distanzmaß zu finden, welches möglichst alle Störfaktoren berücksichtigt, da diese immer von der Umgebung abhängig sind. Sind nun die Distanzen zu mindestens drei Sendern bekannt, so lässt sich daraus mittels Trilateration die Position des Geräts ermitteln. Das Verfahren wurde bereits in 2.3.1.1 ausführlich beschrieben. Ein aus der Praxis entnommenes Beispiel ist der von Apple eingeführte Bluetooth-Lokalisierungsstandard iBeacon[55]. Die dafür eingesetzten Funkbaken werden im Raum an verschiedenen Stellen platziert und senden in regelmäßigen Abständen Funksignale aus, die mit einer eindeutigen Beacon-ID versehen sind. Kommt ein mobiles Gerät in die Nähe der Funkbaken, empfängt es die Funksignale, wertet die jeweiligen Signalstärken aus, errechnet daraus die Distanz und kann schließlich mittels Trilateration die eigene Position bestimmen. Voraussetzung für die Trilateration sind mindestens drei (bei drei Dimensionen vier) Funkbaken die in Reichweite sein müssen.
Neben der Lateration verwendet auch das Tabellen-basierte Verfahren die Signalstärke zur Positionsbestimmung. Allerdings wird im Gegensatz dazu kein Distanzmaß gebildet, sondern lediglich Umgebungsprofile an einer bestimmten Position aufgezeichnet. Als Profilparameter wird die gemessene Signalstärke der entsprechenden Basisstation herangezogen. Die Positionsbestimmung erfolgt dabei über den Vergleich des aktuellen Umgebungsprofils mit den in einer Datenbank gespeicherten Profilen. Das Verfahren ist daher wesentlich unabhängiger von Störfaktoren wie Dämpfung oder Reflexion. Vielmehr macht es sich die Störfaktoren zunutze, da diese das Umgebungsprofil noch weiter charakterisieren, wodurch das Profil für den

[55] Vgl. o. V., iBeacon for Developers - Apple Developer

späteren Vergleich eindeutiger wird. Aus diesem Grund wird die Methode auch oftmals als „Fingerprinting“ bezeichnet, da Fingerabdrücke immer eindeutig sind. Das Fingerprinting besteht aus zwei Phasen: Die Offline-Phase und die Online-Phase. In der Offline-Phase wird zunächst die gesamte Zielumgebung in ein Raster aus mehreren Zellen unterteilt und an jeder Zelle zu jeder Basisstation Umgebungsprofile, sogenannte Fingerprints aufgezeichnet. Solche bestehen in der Regel aus einer eindeutigen ID (z. B. die MAC-Adresse der Sendestation) um die Basisstation eindeutig identifizieren zu können und den dazugehörigen Messdaten. In der Online-Phase werden an einer unbekannten Zelle im Raster Signalstärken in Echtzeit gesammelt. Diese werden mit allen Offline-Referenzdaten verglichen. In welcher Art und Weise die Messdaten gesammelt und schließlich verarbeitet werden, kann nochmals in probabilistische und deterministische Methoden unterteilt werden.
Bei deterministischen Verfahren wird in der Offline-Phase der Median aus allen gesammelten Signalstärken je Basisstation und je Position berechnet und als Tupel in die Datenbank gespeichert. Der Median neutralisiert dabei zufällig aufgetretene Messspitzen und erzeugt daher ein eindeutiges Umgebungsprofil. In der Online-Phase wird jede gesammelte Signalstärke mit den Referenzdaten und einem Vergleichsmaß wie die euklidische Distanz d

$$d_i = \sqrt{\sum_j (s_{ij} - s_{mj})^2} \tag{3}$$

verglichen, wobei s_{ij} die gespeicherte Signalstärke an der Referenzposition i und Basisstation j beschreibt, und s_{mj} die Signalstärke von der selben Basisstation an der Position m darstellt. Ein oft zitiertes Beispiel für ein deterministisches Fingerprint-Verfahren ist das von Microsoft entwickelte RADAR[56]. Besonderheit des Systems ist der Hybridansatz der das Fingerprint-Verfahren mit Trilateration kombiniert, um die Genauigkeit erhöhen zu können. Die Datensammlung und Auswertung erfolgt dabei immer am AccessPoint selbst. Dies erfordert jedoch spezielle Hard- und Software und ist mit handelsüblichen AccessPoints nicht realisierbar. Die Genauigkeit wird mit 5 m angegeben.
Neben deterministischen kommen auch probabilistische Verfahren zum Einsatz. Bei probabilistischen Verfahren werden bei der Offline-Phase nicht der Median bzw. Durchschnittswerte gebildet, sondern für jede Basisstation und jeder Position die Häufigkeitsverteilungen der Signalstärken in der Datenbank gespeichert. Ausgehend von einem neuen, in der Online-Phase aufgezeichneten Vektor S mit den Signalstärken $(s_1, ..., s_k)$ von k AP's wird

[56] Vgl. Bahl und Padmanabhan, RADAR: an in-building RF-based user location and tracking system, S. 3 ff.

dann in der Datenbank eine Position x gesucht, bei der die Wahrscheinlichkeit P mit $P(x \mid S)$ am höchsten ist. Der wohl bekannteste Vertreter für probabilistische Fingerprint Verfahren ist das von Youssef et al. entwickelte System Horus[57]. Es konnte gezeigt werden, dass mit einer probabilistischen Methode in 97% der Fälle eine Genauigkeit von 2 Metern erreicht wird.
Ein großer Nachteil von Fingerprint-Verfahren ist die aufwendige Kalibrierungsphase. Bei den meisten Fingerprint-Systemen hängt die Genauigkeit von der Anzahl der Messdaten ab, die in der Kalibrierungsphase gesammelt wurde. Um die Kalibrierungsphase zu minimieren existieren auch mathematische Modelle, die es erlauben die Signalausbreitungen zu simulieren. Die Schwierigkeit besteht allerdings darin die genannten Störfaktoren im Indoor-Bereich in der Simulation zu berücksichtigen.

2.3.1.5 Bilderkennung

Neben den vorgestellten Funk-basierten Verfahren wird auch Bilderkennung eingesetzt, um die Lokalisierung innerhalb von Gebäuden zu ermöglichen. Solche Systeme benötigen freilich eine Kamera, die heutzutage in jedem mobilen Gerät zu finden ist. Analog zum bildbasierten Tracking im Kontext der Augmented Reality (siehe Kapitel 2.1.2.2), existieren hier wieder verschiedene Methoden um aus Bildern die notwendigen Informationen ziehen zu können. Denkbar sind Marker-basierte Methoden, welche eindeutige Muster aus Bildern erkennen und auswerten können. QR-Codes werden hierfür häufig verwendet, da dafür bereits zahlreiche Implementierungen existieren. Solche Marker werden dann i.d.R. an mehreren Positionen innerhalb eines Gebäudes verteilt und kodieren dann eine eindeutige Position, die durch das Einlesen ermittelt werden kann.
Ferner werden auch markerlose Feature-Matching-Verfahren eingesetzt um eindeutige Merkmale aus dem Bild zu extrahieren und vergleichbar zu machen. Auch hier finden sich Parallelen zu Bild-basierten Tracking-Verfahren, die in 2.1.2.2 beschrieben wurden, sodass diese hier nicht näher erläutert werden. Solche Systeme erfordern, ähnlich wie beim Fingerprint-Verfahren, eine Offline-Phase, in der initial die Bilder aufgenommen und Features im Bild extrahiert und gefiltert werden. In einer Datenbank werden dann die Features mit einer eindeutigen Position gespeichert. In der Online-Phase werden erneut die Features aus dem Live-Bild extrahiert und mit den Features aus der Datenbank verglichen. Als Beispiel aus der Literatur sei hier auf das Verfahren von Kawaji et al.[58] verwiesen, welches sich vor allem

[57] Vgl. Youssef und Agrawala, The Horus WLAN Location Determination System, S.1 ff.

[58] Vgl. Kawaji et al., Image-based Indoor Positioning System: Fast Image Matching Using Omnidirectional Panoramic Images, S.2 ff.

durch hohe Genauigkeit und Verarbeitungszeit auszeichnet.

2.3.2 Wegfindung

Die letzte wichtige Komponente einer Navigation ist die Wegfindung, auch Pathfinding genannt. Dabei geht es um die computergestützte Suche nach dem optimalen Pfad (i.d.R. der kürzeste) von einem Startpunkt zu einem Zielpunkt. Für dieses Problem existieren in der Informatik sogenannte Pathfinding-Algorithmen. Diese bilden Probleme dieser Art als einen gewichteten gerichteten Graphen ab. Dieser Graph G enthält dann mehrere Knoten V und Kanten $E = (v, w)$ mit $v, w \in V$, die v und w miteinander verbinden. Jede Kante hat eine Länge l(v,w)>0, der im Kontext der Navigation die Entfernung von einem Knoten v zu w darstellt. Um nun innerhalb

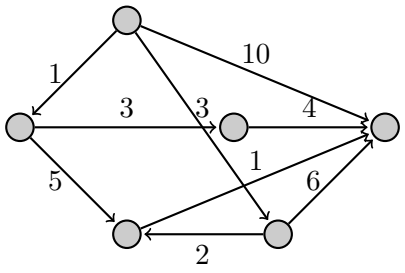

Abbildung 9: Beispiel eines Graphen zur Bestimmung der kürzesten Pfades

des Graphen den kürzesten Pfad von einem Startknoten s zu einem Zielknoten z zu bestimmen, existieren verschiedene Algorithmen. Der wohl am häufigsten verwendete Pathfinding-Algorithmus ist Dijkstra[59]. Dieser beginnt bei einem Startknoten und berechnet sukzessive Knoten für Knoten den kürzesten Weg von s zu allen Nachbarknoten, wobei der kürzeste Weg immer unter Berücksichtigung des Vorgängers $vorg(v)$ und einer bereits erreichten Distanz $dist(v)$ bestimmt wird. Ferner verwendet der Algorithmus zwei Listen: S, um Knoten festzuhalten, von denen der kürzeste Weg von s aus bekannt ist und K, für Knoten, deren Distanz zu s am geringsten ist.

Grundsätzlich ist der Dijkstra-Algorithmus mit einem Aufwand von $O(m \log n)$ relativ effizient. Der Nachteil des Verfahrens ist jedoch, dass der Graph nicht zielführend durchlaufen wird, sondern sich sternförmig ausbreitet. Bei sehr vielen Knoten und Kanten wie bspw. einem Straßennetz im Outdoor-Bereich kann dies hohe Laufzeiten zur Folge haben. Als Erweiterung zum Dijkstra Algorithmus bietet sich daher der A*-Algorithmus an. Dieser erweitert Dijkstra um eine spezielle Heuristik, die zielgerichteter den Graph durchsucht und nur die Knoten betrachtet, die wahrscheinlicher

[59] Vgl. Cormen et al., *Algorithmen - Eine Einführung* -, S.598 ff.

zum Ziel führen.
Die beiden Algorithmen Dijkstra und A* setzen zwingend einen Graph als Datenstruktur voraus um den kürzesten Pfad von einem Punkt zum anderen ermitteln zu können. Im Outdoor-Bereich werden aus Straßennetzen Graphen konstruiert, wobei Kreuzungen als Knoten und Straßen als Kanten mit einer bestimmten Länge abgebildet werden. Zudem steht ein geographisches Koordinatensystem zur Verfügung, um eine Position im freien zu ermitteln. Im Inneren sind die genauen Koordinaten jedoch nicht bekannt und können nur geschätzt werden. Zudem gibt unzählige Möglichkeiten, wie der Benutzer sich im Raum bewegen könnte.

Daher bedient man sich im Indoor-Bereich digitaler Grundrisse eines Gebäudes, um daraus ein Umgebungsmodell konstruieren zu können. Das daraus resultierende Wegenetz kann entweder händisch oder automatisiert durch verschiedene Verfahren erzeugt werden. Beispielhaft zu nennen wären hier die sogenannten „Regular Grids“[60], die Gebäudepläne in Raster unterteilen. Solche Raster bestehen dann aus quadratischen bzw. hexagonalen Feldern, die bis zu 8 Verknüpfungen untereinander erlauben. Jedes Feld stellt dann einen Knoten in einem Graph dar, wogegen jede Verknüpfung zwischen den Feldern eine Kante beschreibt. Eine andere Möglichkeit zur Konstruktion eines Wegenetzes sind Wegpunktgraphen. Solche Wegpunkte können manuell festgelegt, durch Eckpunkte an Hindernissen definiert oder durch Mittelpunkte von Kreisen konstruiert werden, die sich an Flächen bis zum Hindernis aufspannen lassen. Ferner besteht die Möglichkeit, die Gebäudefläche in Polygonen zu unterteilen. Solche Wegenetze bezeichnet man als „Navigation Meshes“. Die Mittelpunkte der Polygone stellen dann die Knoten dar, welche durch Verbinden ein Wegenetz bilden.

2.4 Google Glass

Am 28. Juni 2012 stellte Google erstmals offiziell seine mobile Datenbrille Google Glass auf der jährlich stattfindenden Entwicklerkonferenz Google I/O vor[61]. Das Gerät sollte in der Lage sein, den Computer noch mehr als es bereits mit einem Smartphone der Fall ist, in die reale Welt zu integrieren und die Informationen immer dann bereitzustellen, wenn sie benötigt werden[62]. Damit wagte sich Google in einen Markt, welcher bis dato als Nische galt. Zuvor hatte es viele gescheiterte Ansätze gegeben eine alltagstaugliche Datenbrille zu etablieren. Und obwohl die Brille nur als Explorer Edition für Test- bzw. Entwicklungszwecke herausgegeben wurde, gewann sie hohe mediale Aufmerksamkeit.

[60] Vgl. Remus, Fachhochschule Wedel – Informatikseminar SS2007 2007, S.671 ff.
[61] Vgl. Volpe, Sergey Brin demos Project Glass onstage at Google I/O
[62] Vgl. Brin, Sergey Brin: Why Google Glass?

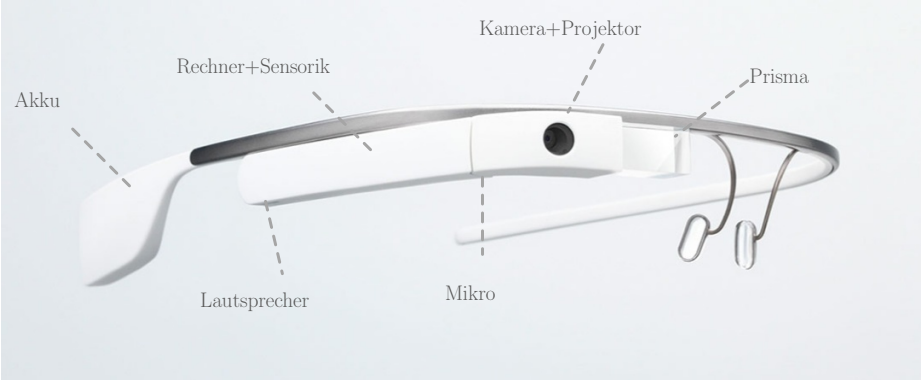

Abbildung 10: Die Google Glass. [63]

Die Google Glass besteht aus einem Miniaturcomputer, der auf der rechten Seite eines Brillenrahmens montiert ist. Der Computer wird ähnlich wie bei einem Smartphone durch Sensoren wie einer Kamera, Inertialsensorik, Helligkeits- und Näherungssensoren ergänzt. Ferner beinhaltet die Datenbrille eine WLAN- und Bluetooth-Schnittstelle, um die Kommunikation mit dem Internet und dem Smartphone zu ermöglichen. Der Akku befindet sich am rechten Ohrbügel, die Audio-Ausgabe erfolgt über einen Knochenleitungslautsprecher. Besonders aus technischer Sicht hervorzuheben ist das im oberen rechten Sichtbereich eingesetzte Prisma, das in Kombination mit einem Projektor das virtuelle Bild produziert.

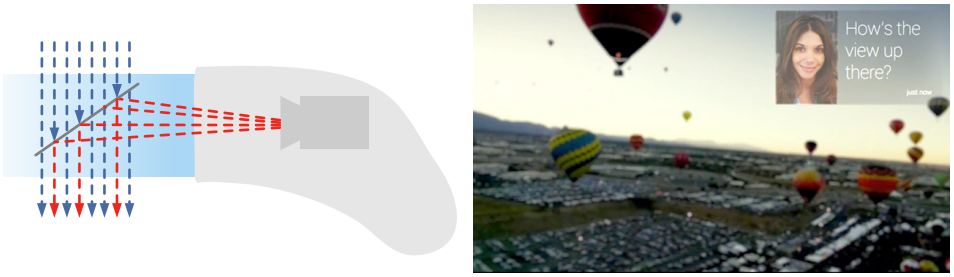

Abbildung 11: Glasprisma der Google Glass (links)[65]und virtuell erzeugtes Bild (rechts).

Das Prisma enthält einen semi-transparenten Spiegel, der die Lichtstrahlen eines Projektors in einem bestimmten Winkel so spiegelt, dass diese direkt in die Retina des Auges projiziert werden, um scharfes sehen auf naher Sicht zu ermöglichen[66]. Ferner wirkt der Spiegel als Filter für natürliches Licht, sodass dieses das künstlich erzeugte Licht aus dem Projektor nicht überblendet. Das virtuell erzeugte Bild sieht der Benutzer nur scharf, wenn dieser leicht in den oberen rechten Bereich blickt. Die Bedienung der

[63]Siimon, Why Google Glass Failed: A Marketing Lesson
[65]Vision, Google Glass - Was ist das eigentlich?
[66]Vgl. a. a. O.

Brille erfolgt entweder über Kopfbewegungen, Sprachbefehle oder mit dem im Gehäuse befindlichen Touchpad. Ferner ist ein Eye-Tracker integriert, welcher die Steuerung mittels Zwinkern ermöglicht.
Das Betriebssystem der Google Glass basiert auf Android und profitiert damit von dessen Abstraktionsmodell. Android stellt einen vollständigen Architektur-Stapel von hardwarenaher Treiber-Programmierung bis hin zum Applikations-Framework bereit, dazwischen Schichten für das Betriebssystem, native Bibliotheken, Middleware und eine Java Laufzeitumgebung auf welcher das Android-SDK (Software Development Kit) zum Entwickeln von Applikationen aufsetzt. Die Art der Interaktion und User Experience mit der Google Brille unterscheidet sich jedoch stark von einem handelsüblichen Smartphone, sodass Android entsprechend modifiziert werden musste. So ist das am Bügel befestigte Touchpad kein Display, welches man bedient. Zudem ist dieses relativ schmal, sodass es nur drei Arten von Interaktionen erlaubt: Das Wischen nach links und rechts, um einen Kontextwechsel durchzuführen, das Wischen nach unten, um zum vorherigen Kontext zurückzukehren und schließlich das Auswählen durch leichtes antippen. Daher wird statt einem Dialogfeld mit mehreren Eingabekomponenten immer nur eine Liste von Dialogen angeboten. Diese Liste wird Timeline genannt und stellt die Hauptbenutzerschnittstellenkomponente der Google Glass dar[67]. Sie besteht aus mehreren 640x360 Pixel großen Karten (sogenannten Cards), durch welche der Benutzer navigieren kann. Es werden grundsätzlich zwischen den folgenden Arten von Cards unterschieden:

1. Static Cards - Visualisieren Informationen die sich nicht, oder nur sehr selten ändern z.B. ein Bild mit einer Beschreibung.

2. Live Cards - Stellen Echtzeit-Informationen dar, die zur Gegenwart oder in der Zukunft relevant sein könnten z.B. ein Aktienkurs.

3. Settings - Diese Cards ermöglichen die Google Glass zu konfigurieren.

4. Immersion - Sogenannte Immersions sind eine Menge von Cards, die unabhängig von der Timeline als eigene Glassware auftreten z.B. eine Lagerverwaltungsapplikation.

5. Menu Items - Jede Immersion oder Card kann ein Menü beinhalten mit welchen Aktionen wie bspw. Teilen oder Löschen durchgeführt werden können.

Die Timeline ist in mehreren Abschnitten organisiert (siehe Abb. 12). Im linken Bereich befinden sich die Cards, die Echtzeit-relevante Informationen und Informationen zu anstehenden Ereignissen visualisieren. Im rechten

[67] Vgl. o. V., Google Developers

Bereich werden Cards gelistet, die in der Vergangenheit aufgetreten sind. Diese werden dabei nach Datum des Auftretens sortiert. Beide Abschnitte werden durch eine Home-Karte getrennt. Diese Card ist der Einstiegspunkt der Google Glass und erlaubt es mittels Sprache und Touch-Gesten neue Glassware auszuführen.

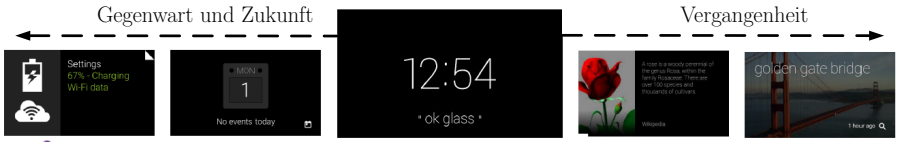

Abbildung 12: Die Google Glass Timeline

Es gibt prinzipiell drei Arten von Glassware und damit auch drei Möglichkeiten solche zu entwickeln[68].
Bei der ersten Variante werden Web-basierte Dienste über eine Cloud-basierte REST-API[69] , die sogenannte Mirror API entwickelt. Der Quellcode befindet sich dabei nicht auf dem Gerät sondern auf einem Server, der mit der Google Glass kommuniziert. Dies lässt sich mit Webseiten im World Wide Web vergleichen. Die Vorteile dieser Variante ist die Plattform- und Programmiersprachenunabhängigkeit, und die Möglichkeit eine gemeinsame Infrastruktur zu nutzen. Nachteilhaft ist zum einen, dass die API dauerhaft einen Internetzugang voraussetzt. Zum andern, dass sich die Hardware, wie Sensoren, Kamera etc. nicht direkt von der REST-API ansprechen lässt. Will man also native und unter Umständen Offline-basierte Apps auf der Google Glass entwickeln, benötigt man das Glass Development Kit (kurz GDK). Bei dieser Variante lassen sich Apps für die Google Glass entwickeln, die direkt auf dem Gerät ausgeführt werden. Grundsätzlich setzt das GDK auf das Android SDK auf, wobei es dieses um die Standardkomponenten, wie Cards und Immersions oder Spracherkennung und Wischgestenerkennung erweitert. Ansonsten steht dem Entwickler derselbe Umfang an Bibliotheken bereit, um Applikationen zu entwickeln. Als dritte Variante, hat man als Entwickler die Möglichkeit, die Vorteile beider Varianten kombinieren, in dem einen hybriden Ansatz wählt.

[68] Vgl. o. V., Google Developers
[69] Representational State Transfer - Zustandslose, auf HTTP basierende Webservices

3 Analyse und Konzept

Im folgenden Kapitel sollen aufbauend auf den Grundlagen, die Analyse-Phase und der konzeptuelle Teil der Arbeit beschrieben werden. Dabei werden ausgehend von den gegebenen Anforderungen mögliche Verfahren evaluiert und auf ihre Machbarkeit untersucht. Schließlich soll ein Konzept für eine Optical-See Through basierte Augmented Reality Navigationsapplikation entstehen, welches die Basis für die Implementierung stellt.

3.1 Anforderungsanalyse

Eines der Hauptziele der Arbeit, ist den Hauptnutzen der erweiterten Realität, nämlich die Minimierung der Zeit zur Informationsbeschaffung, anhand eines Beispielszenarios zu demonstrieren. In erster Linie soll daher der konkrete Anwendungsfall der Technischen Hochschule Köln untersucht werden. Man stelle sich grundsätzlich zwei mögliche Szenarien vor:

1. Ein Student oder Dozent befindet sich in einem bestimmten Flur des Gebäudes und sucht einen freien Raum.

2. Ein neuer Student, Dozent oder Besucher möchte eine bestimmte Vorlesung besuchen bzw. sucht einen spezifischen Raum innerhalb des Gebäudes.

Im ersten Fall wäre es erforderlich, dass der Benutzer sich vorher informiert, welche Räume frei sind oder jeden Raum von Tür zur Tür öffnet um einen freien Raum auszumachen. Um an dieser Stelle die Informationsbeschaffung auf Abruf zu ermöglichen und damit die Minimierung der Zeit zur Informationsbeschaffung zur reduzieren, wäre es von Nutzen, wenn jeder Raum eine virtuelle Beschilderung, wie in Abbildung 13 dargestellt erhielte, die dem Nutzer Informationen zu diesem visualisiert. Ein solcher Raum würde dann mit einer eindeutigen Raum-Nr. und der aktuellen und nächsten Vorlesung beschildert werden. Auf diese Weise könnte der Nutzer erkennen, ob es sich um einen freien oder besetzten Raum handelt.
Im zweiten Fall möchte der Benutzer von einer gegebenen Position zur nächsten navigieren. Auch hier hätte er die Möglichkeit sich vorher zu informieren, einen Raumplan auszudrucken, Personen im Gebäude zu befragen usw. Um jedoch erneut auf Anforderung und minimalem Zeitaufwand die Navigation innerhalb des Gebäudes zur ermöglichen, wäre eine IT gestützte Wegfindung denkbar. Dabei könnte der Benutzer ein Ziel wie bspw. einen Raum auswählen und die Anwendung den Benutzer zum gewünschten Flur navigieren. Durch die Anzeige der Rauminformationen in seiner Umgebung findet er zudem den gesuchten Raum und erhält dazu Informationen zur

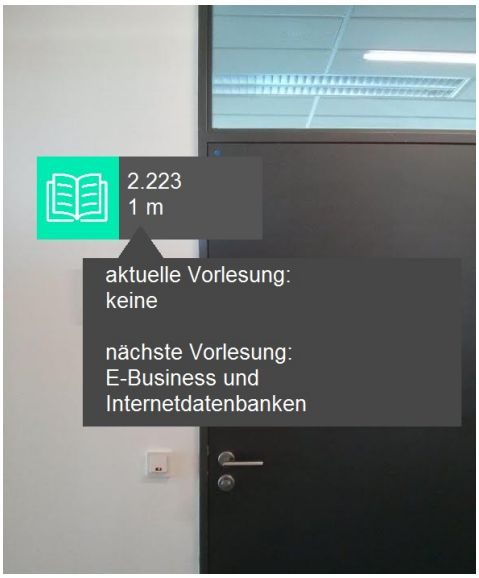

Abbildung 13: Kontextabhängige Informationen als Erweiterung der Realität

aktuellen und nächsten Vorlesung.

Durch Nutzung der Datenbrille Google Glass soll der Nutzeffekt noch weiter verstärkt werden. Bei einem Smartphone wäre es erforderlich zunächst die App zu starten, das gewünschte Ziel einzugeben und zu bestätigen. Mithilfe der Google Glass lässt sich die Zeit zur Informationsaufnahme durch Sprachbefehle noch weiter reduzieren. So könnte durch einen Befehl wie „Navigiere zum Raum X“ bereits die Navigation angestoßen werden. Ferner sollen virtuelle Informationen mittels Optical-See-Through unmittelbar und in Echtzeit visualisiert werden. Wenn der Benutzer also den Flur betritt, werden diesem beim Blick auf die Türen der Räume umgehend Kontext-abhängige Informationen zu diesen eingeblendet. Ändert sich der Raum einer Vorlesung, wird der Benutzer dennoch zum richtigen Ziel geleitet. Auf diese Weise erhält der Benutzer immer die benötigten Informationen zur richtigen Zeit am richtigen Ort. Letztlich gilt es sicherzustellen, dass die Benutzerschnittstelle intuitiv bedienbar ist und der Benutzer durch möglichst kurze und prägnante Informationen nicht unter Informationsüberflutung leidet.

3.2 Wahl der AR Methodik

Aus der Anforderung heraus, eine Augmented Reality Applikation zur Navigation innerhalb von Gebäuden zu implementieren, ergeben sich zahlreiche Realisierungsmöglichkeiten. Die in den Grundlagen beschriebenen Verfah-

ren zur Positionsbestimmung haben bereits verwandte Arbeiten aufgezeigt, die reine Indoor-Navigation ermöglichen. Ferner wurde gezeigt, welche technischen Implementierungsmöglichkeiten für eine Augmented Reality-App existieren und dessen Ausprägungen auf mobilen Geräten vorgestellt. Einen Anwendungsfall, bei welchem beide Anwendungsbereiche im Indoor-Bereich kombiniert und mittels Optical-See-Through realisiert werden, existiert in der Literatur allerdings nicht. Die besondere Herausforderung besteht also darin unter Berücksichtigung der gegebenen Anforderungen, die beiden Anwendungsfelder der Indoor Navigation und Augmented Reality zusammenzuführen.

In den Grundlagen wurde bereits zwischen objektbasierter und ortsbasierter AR unterschieden. Im ersten Schritt gilt es zu untersuchen, welche Methode sich für die Umsetzung der Indoor-AR-Navigation mit der Google Glass am ehesten eignet. Wie im Kapitel 2.3 beschrieben, definiert sich die Navigation aus den Aspekten der Wegfindung und Orientierung im topographischen Raum. Für eine optimale Wegfindung ist ein räumlicher Bezug und daher eine Positionsbestimmung unabdingbar. Es liegt nahe, dass durch die ortsbasierte AR bereits die notwendigen Voraussetzungen gegeben sind, da diese ebenfalls auf Positionsbestimmung basiert, während für die objektbasierte Variante zusätzlicher Aufwand notwendig ist, um diese zu realisieren. In dieser Hinsicht lassen sich also durch Nutzung der ortsbasierten AR bereits Synergieeffekte erzeugen.

Die Orientierung im topographischem Raum, soll in dieser Arbeit durch virtuelle Informationserweiterung erreicht werden. Auf diese Weise wird die Verknüpfung zur erweiterten Realität hergestellt. Für den beispielhaften Anwendungsfall an der Technischen Hochschule am Campus Gummersbach, erfolgt hierfür die Informationserweiterung durch virtuelle Beschilderungen von Räumen, die am Türbereich platziert werden. Virtuelle Beschilderungen lassen sich zwar mit objektbasierten Verfahren realisieren, haben jedoch den Nachteil, dass sie auf eindeutige und vergleichbare Merkmale im Bild angewiesen sind. Hier wurden während der Evaluationsphase bereits mögliche Verfahren erprobt, um eindeutige Features aus Bildern von Fluren und Türen des Gebäudes zu extrahieren. Man konnte jedoch feststellen, dass aufgrund des recht ähnlichen Aussehens der Flure und Türen, die Features nicht eindeutig genug waren, um einen Vergleich zu ermöglichen. Ein weiterer Nachteil solcher Verfahren ist die hohe Anforderung an der Rechenleistung. Es konnte gezeigt werden, dass die Bildwiederholungsrate auf einem handelsüblichen Smartphone aufgrund des Mustererkennungsalgorithmus deutlich reduziert wurde. Daher ist anzunehmen, dass die virtuelle Erweiterung der Realitätswahrnehmung stark unter Schwimmeffekten lei-

den würde.
Eine Alternative sind Marker, die an jeder Tür platziert werden. Solche Marker sind in der Lage eine eindeutige ID des Raumes kodieren, mit welchem man Informationen aus einer Datenbank abrufen kann. Diese Bildbasierte Variante lässt sich mit geringen Kosten und Aufwand mit gedruckten Markern realisieren. Zudem werden die virtuellen Inhalte exakt an der Stelle des Markers platziert, und damit eine hohe Genauigkeit erreicht. Ein solches Verfahren wurde daher während der Evaluationsphase an einem existierenden Marker-Tracker erprobt. Es wurde deutlich, dass für eine Entfernung von bis zu 3 Metern mindestens ein Marker mit der Große von 25x25 cm notwendig wäre, um das eindeutige Muster des Markers aus dem Bild extrahieren zu können. Folglich wäre die erweiterte Realität nur auf kurzer Distanz realisierbar, was für den konkreten Anwendungsfall nicht in Frage kam.
Ein ortsbasiertes System dagegen, bezieht die Informationen aus dem räumlichen Kontext. Auf diese Weise können auch Informationen im Indoor-Bereich aus größerer Entfernung und sogar ohne direkten Sichtkontakt(z. B. innerhalb eines Flurs) visualisiert werden. Dies ist ein entscheidender Vorteil der ortsbasierten Methode. Ein solches System erfordert allerdings ein Umgebungsmodell und die Möglichkeit, die Position des Benutzers zu ermitteln. Letzteres stellt im Indoor-Bereich eine besondere Herausforderung dar, müsste allerdings für eine Indoor-Navigation ohnehin realisiert werden. Der Unterschied liegt allerdings im Anspruch auf Genauigkeit. Für das Navigationssystem reicht eine Genauigkeit von 3-5 Metern zur Wegfindung aus. Die AR-Komponente muss allerdings mindestens eine Genauigkeit von 1-2 Metern aufweisen, damit virtuelle Inhalte nicht vollständig verrutscht platziert werden.
Die Synergieeffekte, die durch ein ortsbasierendes System mit Navigation erreicht werden können, die Fähigkeit, auch aus größerer Entfernung AR zu ermöglichen und schließlich die geringere Anforderung an Rechenleistung waren schließlich ausschlaggebend ein ortsbasiertes System als Grundlage für die Entwicklung einer Indoor-basierten AR Navigationsanwendung zu nutzen.

3.3 Grundkonzept

Angesichts der Entscheidung ein ortsbasierendes AR-System im Indoor-Bereich zu entwickeln, ergeben sich einige Grundvoraussetzungen. Die grundlegende Funktionsweise eines ortsbasierenden AR-Systems wurde in Kapitel 2.2.2 bereits beschrieben. Einen Ansatz ortsbasierte AR im Indoor Bereich zu realisieren wurde in der Literatur nicht gefunden. Prinzipiell sollten jedoch die gleichen Voraussetzungen gelten wie im Outdoor-Bereich. In erster

Linie geht es darum zu ermitteln, wo sich der Benutzer aktuell aufhält (Wo bin ich?). Dafür muss ein Koordinatensystem bekannt sein, in welchem relative Koordinaten bestimmt werden können. Ist die Position bekannt, können Standort-bezogene Informationen ermittelt werden (Was ist in meiner Nähe?). Hierfür müssen vorher Informationen im Koordinatensystem hinterlegt sein. Im letzten Schritt kommt der AR-Aspekt hinzu. Hierbei geht es darum Kontext-abhängige Informationen zu visualisieren, also zu ermitteln welche Informationen im Blickfeld eingeblendet werden sollten (Was sehe ich?). Um alle genannten Aspekte zu realisieren wurde ein Grundkonzept erarbeitet, welches im Folgenden erläutert wird.

3.3.1 Aufbau eines Koordinatensystems

Der erste Prozessschritt setzt voraus, dass ein räumliches Koordinatensystem existiert, in welches Informationen an ausgewählten Punkten hinterlegt werden können. Im Zusammenhang mit GPS wird im Outdoor-Bereich ein geographisches Koordinatensystem aus den Informationen Längen- und Breitengrad genutzt, um die Lage eines Punktes auf der Erde beschreiben zu können. Im Indoor-Bereich wären geographische Koordinaten zwar denkbar, jedoch deutlich aufwendiger, da die geographischen Koordinaten zunächst ermittelt werden müssten, bevor ein Umgebungsmodell konstruiert werden kann. Dies wäre wiederum nur theoretisch möglich, da die genaue geographische Position nicht ermittelbar ist. Daher müsste man wahrscheinlich auf virtuelle Koordinaten zurückgreifen. Stattdessen wurden für diese Arbeit relative Koordinaten verwendet. Um ein System zu finden, welches zusätzlich den Anforderungen der Indoor-Navigation genügt, wurde zudem ein entsprechendes Umgebungsmodell konstruiert. Dabei wurde ein bestehender Gebäudeplan mit einem einfachen Regular Grid, bestehend aus quadratischen Feldern versehen, welches die Navigation in 4 Richtungen erlaubt.

Das Raster beschreibt ein zweidimensionales kartesisches Koordinatensystem P aus $n \cdot m$ Punkten (x_1, x_2). Der Ursprung O(0,0) des Koordinatensystems liegt im oberen linken Bereich (siehe Abb. 14), die Y-Achse ist daher gespiegelt. Die Positionsbestimmung muss dann immer einen spezifischen Punkt innerhalb des Koordinatensystems zurückgeben. Dabei gilt $x_1 \in \mathbb{N}$ und $x_2 \in \mathbb{N}$. Die Positionierung erfolgt also diskret in einem Raum von natürlichen Zahlen. Zudem wird jeder Raum an einer spezifischen Koordinate mit Informationen versehen. Solche Standortinformationen werden im Kontext der ortsbasierten AR auch Points of Interests (POIs) genannt. Jeder POI kann ein Raum, ein Aufzug, eine Treppe etc. sein und dient dazu, dem Benutzer die Orientierung an seiner aktuellen Position zu ermöglichen. Im Rahmen dieser Arbeit, werden lediglich Vorlesungsräume als POIs be-

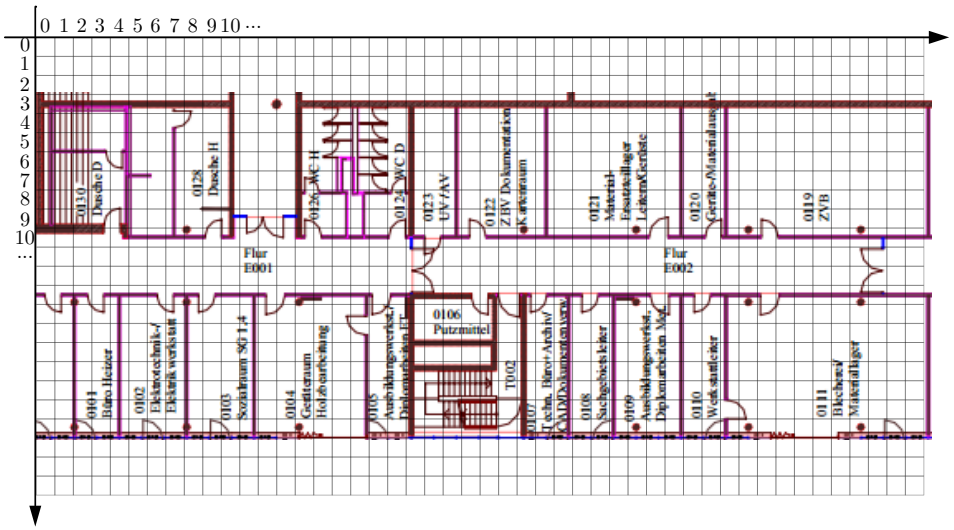

Abbildung 14: Regular Grid für ein Indoor Umgebungsmodell

trachtet.
Ein dreidimensionales Koordinatensystem wäre eine sinnvolle Erweiterung, um mehrere Etagen des Gebäudes abzubilden. Diese Arbeit beschränkt sich allerdings auf den zweidimensionalen Fall. Um mehrere Etagen abzubilden, müssen daher mehrere Karten pro Etage erzeugt werden. Dies hat den Nachteil, dass Übergänge zwischen Karten, wie etwa Treppen bzw. Aufzüge nicht abbildbar sind. Da jedoch in einem Treppenhaus oder Aufzug keine POIs zu erwarten sind, beeinträchtigt dies nicht die Funktionalität des ortsbasierten AR Systems. Eine zweidimensionale Karte ist daher für diesen Anwendungsfall ausreichend.

3.3.2 Positionsbestimmung

Die wichtigste Komponente eines ortsbasierten AR-Systems ist die Positionsbestimmung. Wie im Kapitel 2.3.1 beschrieben, stellt die Positionsbestimmung im Indoor-Bereich eine besondere Herausforderung dar, da höhere Anforderungen an die Genauigkeit und Wiederholrate gestellt werden. Ferner kann GPS nicht genutzt werden, da die meisten GPS-Signale vom Gebäude gedämpft oder reflektiert werden. Im Kapitel wurden zudem mögliche Verfahren vorgestellt, mit welchen Indoor-Positionsbestimmung ermöglicht werden kann. Während der Konzeptphase wurden hieraus drei mögliche Verfahren zur Evaluation ausgewählt. Im Folgenden sollen diese gegenübergestellt werden.

3.3.2.1 WLAN Fingerprinting

Das Fingerprinting-Verfahren wurde in Kapitel 2.3.1 eingeführt. Im konkreten Fall von WLAN stellen die sogenannte Basisstationen AccessPoints

(AP's) dar. Jeder AP sendet in regelmäßigen Abständen einen Beacon-Frame, damit umgebende Geräte den AP erkennen und eine anfängliche Verbindung mit diesem herstellen können. Ein solcher Beacon-Frame enthält unter anderen eine eindeutige SSID (i.d.R. die MAC-Adresse des AP's), einen Timestamp und das Intervall in welchem der AP einen Beacon sendet. Die Signalstärke wird am Empfangsgerät gemessen.
Wie bereits in den Grundlagen behandelt, kann mit einem probabilistischen Verfahren prinzipiell eine höhere Genauigkeit der Positionsbestimmung erreicht werden. Deterministische Verfahren haben den Nachteil, dass durch die Mittelung der Signalstärken u. U. charakteristische Merkmale verloren gehen. In dieser Arbeit wird daher ein probabilistischer Ansatz nach Youssef et al.[70] gewählt.
Dabei werden in der Offline-Phase an q Referenzpunkten $P = \{p_1, ..., p_q\}$ im erzeugten Raster, die zu jedem Referenzpunkt zugehörigen Messwerte $M_j = \{m_1, ..., m_k\}$, $j \in \{1, ..., q\}$ als Stichproben mit k möglichen AP's in einer Datenbank gesammelt. Zu jeder Stichprobe $m_i = (s_i, w_i)$, $i \in \{1, ..., k\}$ wird jeweils die empfangene Signalstärke s_i und eine eindeutige MAC-Adresse w_i gespeichert. Die MAC-Adresse ist eine aus dem Beacon-Frame entnommene 48-Bit große Identifikation, die vom Hersteller vorgegeben ist und sich i.d.R. nicht ändert. Sie dient zur eindeutigen Identifizierung des AP's und vereinfacht die spätere Zuordnung. Sie wird als RSSI-Wert in dBm ausgedrückt, wobei das angegebene Messintervall vom Hersteller vorgegeben wird. Es wird von einem Intervall zwischen [-100;0] ausgegangen, wobei 0 eine optimale Empfangsleistung ausdrückt. Es gilt daher $s_i \in \{-100, ..., 0\}$.
In der Online-Phase wird an einer unbekannten Position j eine Stichprobe M mit $(m_1, ..., m_k)$ für k AP's gesammelt. Ziel eines probablistischen Fingerprint-Verfahren ist es, für p_j die Position zu finden, die die Wahrscheinlichkeit für das Auftreten der Stichprobe M maximiert[71]. Nach dem Satz von Bayes lässt sich dies folgendermaßen formulieren[72]:

$$\arg\max_{p_j}(P(p_j \mid M)) = \arg\max_{p_j}(\frac{P(M \mid p_j) \cdot P(p_j)}{P(M)}) \tag{4}$$

P(M) ist bei Variation von j immer konstant und spielt für die Berechnung des argmax Ausdruck daher keine Rolle. Man erhält also:

$$\arg\max_{p_j}(P(p_j \mid M)) = \arg\max_{p_j}(P(M \mid p_j) \cdot P(p_j)) \tag{5}$$

[70] Vgl. Youssef und Agrawala, The Horus WLAN Location Determination System, S. 1 ff.
[71] Vgl. a. a. O., S.3
[72] Vgl. a. a. O., S.5

Da außerdem das nötige Vorwissen fehlt um die a-priori-Wahrscheinlichkeit für das Auftreten eines Referenzpunkts $P(p_j)$ zu bestimmten, muss eine Gleichverteilung angenommen werden. Letztlich muss also nur $P(M \mid p_j)$ betrachtet werden. Daraus entsteht die Maximum-Likelihood-Hypothese:

$$\arg\max_{p_j}(P(p_j \mid M)) = \arg\max_{p_j}(P(M \mid p_j)) \tag{6}$$

Youssef et al. treffen dabei die Annahme, dass die Signalstärken der AP's voneinander unabhängig sind, da sich in einem gut konzipierten WLAN-Netzwerk die WLAN Kanäle grundsätzlich untereinander nicht überlappen und somit auch gegenseitig nicht stören. Aufgrund der Unabhängigkeit der auftretenden Signalstärken, lässt sich die Gesamtwahrscheinlichkeit $P(p_j \mid M)$ für eine Position unter Annahme einer empfangenden Stichprobe M für jeden in der Offline-Phase gespeicherten Messpunkt j folgendermaßen bestimmen:

$$P(M \mid p_j) = \prod_{i=1}^{k} P(m_i \mid p_j) = \prod_{i=1}^{k} P((s_i, w_i) \mid p_j) \tag{7}$$

$P((s_i, w_i) \mid p_j)$ bezeichnet dabei die Einzelwahrscheinlichkeit für das auftreten eines Messwertes (s_i, w_i) an der Position p_j. Ein probabilistisches Fingerprint-Verfahren speichert also nicht je AP und Position die gemittelte Signalstärke, sondern ermittelt für jede Signalstärke die relative Häufigkeit. Dies kann entweder die tatsächliche Häufigkeit, oder ein durch eine Funktion approximierter Wert sein. Die Häufigkeitsverteilung wird klassischerweise in einem Histogramm visualisiert. Abbildung 15 veranschaulicht ein Histogramm für einen AP an einer gemessenen Position.

Das Histogramm wurde aus 20 gesammelten Signalstärken-Stichproben eines AP's gebildet, die an einer festen Position ohne Bewegung des Empfangsgerätes entnommen wurden. Man kann feststellen, dass bereits bei einer sehr kleinen Stichprobenmenge die Signalstärken stark abweichen bzw. schwanken. Diese Schwankungen ergeben sich aus den in dem vorherigen Kapiteln erwähnten Mehrwegeausbreitung der Funksignale. In dem obigen Beispiel streuen sich die Signalstärken in einem Intervall von [-84;-77] mit einer Standardabweichung von 1,74.
Aus dem Histogramm der Abbildung 15 lässt sich vermuten, dass die Häufigkeitsverteilung bei wenigen Stichproben durch eine Normalverteilung beschrieben werden könnte. Aus dem Quantil-Quantil-Plot im rechten Schaubild lässt sich schließen, dass die Quantile der beobachteten Werte ungefähr mit den Quantilen der Normalverteilung korrelieren. Eine Normalverteilung hätte den Vorteil, dass die relativen Häufigkeiten zur Laufzeit nicht bekannt sein müssten, um die Wahrscheinlichkeit für das Auftreten einer Signalstär-

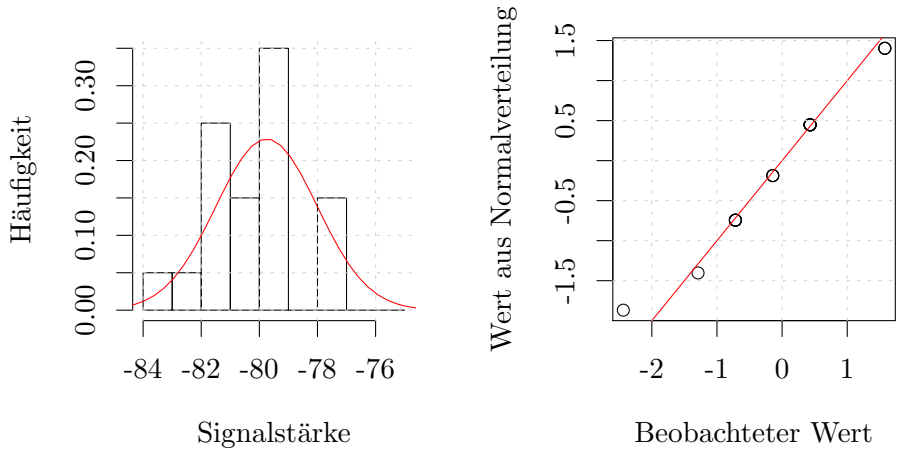

Abbildung 15: Häufigkeitsverteilung eines AP's an einer Position

ke abzuleiten. Somit könnten die Einzelwahrscheinlichkeiten mittels einer Normalverteilungsfunktion zur Laufzeit berechnet werden.
Betrachtet man jedoch Abbildung 16, wird deutlich, dass durch Erhöhung der Stichproben die Annahme einer Normalverteilung nicht mehr zutrifft. Hierbei wurden 1576 Signalstärke-Stichproben eines AP's an einer Position aufgenommen. Besonders auffällig ist hierbei die höhere Streuung (im Intervall [-46;-30]) und die Mehrfachspitzen. Zudem konnten Lücken beobachtet werden, in denen Signalstärken gar nicht oder nur in seltenen Fällen auftreten, wie etwa bei der Signalstärke -36. Sollte diese in der Online-Phase einmal auftreten, würde die Normalverteilungsfunktion eine stark abweichende Wahrscheinlichkeit zurückliefern, was wiederum zur Fehlpositierung führen würde. Zwar wurden in der Literatur zur Approximation zweier Normalverteilungen bei doppelten Spitzen bereits Ansätze vorgestellt[73], jedoch müssen hierbei verschiedene Fälle, wie etwa Ein- oder Mehrfachspitzen berücksichtigt werden, die die Komplexität des Systems weiter erhöhen. Aufgrund dieser Unzulänglichkeiten wäre eine Approximation durch eine Normalverteilung daher nur mit höherem Optimierungsaufwand vertretbar. Weitere Möglichkeiten zur Approximation werden im Rahmen in dieser Arbeit nicht weiter verfolgt.

Stattdessen werden die tatsächlich aufgetretenen relativen Häufigkeiten betrachtet. Dabei wird nach der Offline-Phase für jeden AP und jede Position ein Histogramm, wie in Tabelle 1 dargestellt, generiert. Das bedeutet auch, dass für jede mögliche Signalstärke, je Position und AP eine Wahr-

[73] Vgl. Chen et al., Sensors, Nr. 8, Bd. 13, 2013, S. 3 ff.

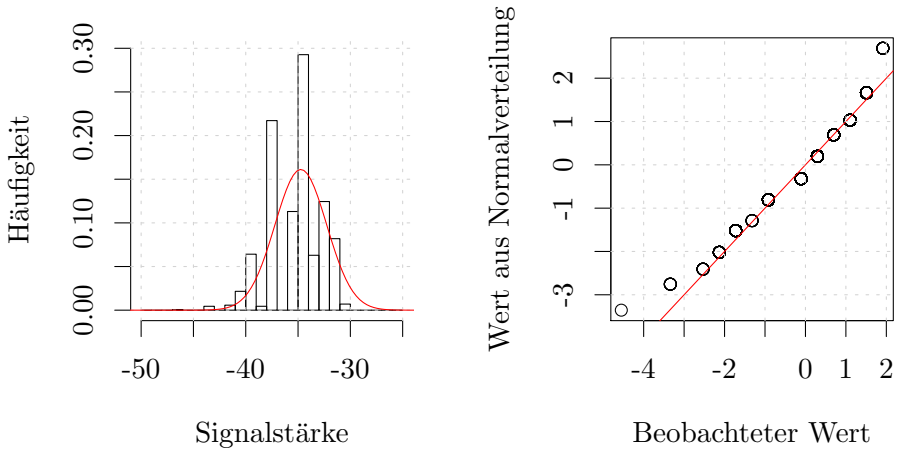

Abbildung 16: Beispiel-Histogramms eines AP's, dem die Verteilung der Signalstärken von der Normalverteilung abweichen

Feld	AP	Signalstärke	Abs. Häufigkeit	Rel. Häufigkeit
		...		
101	4	-77	1	0.027
101	4	-76	1	0.027
101	4	-75	0	9.80909e-45
101	4	-74	0	9.80909e-45
101	4	-73	2	0.0541
101	4	-72	8	0.2162
101	4	-71	2	0.0541
101	4	-70	1	0.027
		...		

Tabelle 1: Gespeichertes Histogramm für einen AP 4 an der Position 101

scheinlichkeit gespeichert werden muss. Zwar ist hierfür ein deutlich höherer Speicherplatzbedarf notwendig, jedoch ist bei der Abfrage der Wahrscheinlichkeiten der Rechenaufwand geringer, da die Wahrscheinlichkeiten bereits vorliegen. So kann beispielsweise die Wahrscheinlichkeit für das Auftreten der Signalstärke -73 an Position 101 und AP 4 also $P((-73, 4) \mid 101) = 0.0541$ direkt vom Histogramm ausgelesen werden. Die Berechnung der Gesamtwahrscheinlichkeit einer Stichprobe erfolgt dann nach der Formel (7). Man erkennt zudem an Tabelle 1, dass die Wahrscheinlichkeiten mit einem initialen Schwellwert $\lambda = 9.80909e - 45$ belegt werden. Es besteht das Risiko, dass AP's in Stichproben fehlen durch den Ausfall eines AP's. Ist dies der Fall, wird grundsätzlich immer von einer Einzelwahrscheinlichkeit $P((s_i, w_i) \mid p_j) = 0$ ausgegangen, wodurch die Gesamtwahrscheinlichkeit

durch die Multiplikation der Einzelwahrscheinlichkeiten stets 0 ergibt. Die Vergleichbarkeit ist dann nicht mehr gegeben und verfälscht die Ergebnisse. Daher wird ein hinreichend kleiner Schwellenwert λ definiert, der zu allen Wahrscheinlichkeiten hinzuaddiert wird, sodass in der Gesamtheit die Proportionen gleich bleiben und immer gleich viele Einzelwahrscheinlichkeiten aggregiert werden.

Zur Evaluation des Fingerprintings, wurde eine Testumgebung mit 12 Positionen konstruiert, die jeweils 1 Meter voneinander entfernt waren. An jeder Position wurden in der Offline-Phase 30 WLAN-Scan-Durchläufe (4 Sekunden pro Scan) aufgezeichnet und in der Online-Phase 16 (4 pro Test). Die Beacons wurden mit einem handelsüblichen Smartphone (Galaxy SII) aufgezeichnet. Da das Gerät dasselbe Funkmodul (BCM4330), wie die Google Glass verbaut hat, war dies ein ideales Testszenario. Um eine Vergleichbarkeit mit anderen Testergebnissen zu ermöglichen, wurden stets die gleichen Voraussetzungen geschaffen. So wurden die Stichproben nur in einer Richtung und immer an den selben Stellen entnommen. Zudem wurden Tests mit gleichen Daten am selben Tag und unmittelbar hintereinander durchgeführt. Die Testergebnisse werden in Tabelle 2 zusammengefasst.

(a)

(b)

Pos	**T1**	**T2**	**T3**	**T4**	∅
1	0,75	0,00	1,00	0,25	0,50
2	0,75	1,75	0,25	0,25	0,75
3	1,00	0,75	0,00	0,75	0,63
4	0,25	0,50	0,75	0,75	0,56
5	2,00	0,00	0,50	1,75	1,06
6	0,25	1,25	0,75	0,50	0,69
7	0,50	1,75	1,50	2,00	1,44
8	0,50	1,50	0,50	2,00	1,13
9	0,50	0,25	0,00	0,00	0,19
10	0,75	0,50	1,00	1,00	0,81
11	0,25	0,25	0,75	1,00	0,56
12	1,00	0,75	0,50	0,25	0,63

Tabelle 2: Versuchsaufbau (a) und Versuchsergebnisse - Fehler in m(b)

Die Tabelle zeigt die durchschnittlichen Fehler pro Testdurchgang für jede Position und der durchschnittlichen Fehler pro Position insgesamt. Der Fehler berechnet sich anhand der euklidischen Distanz zwischen der tatsächlichen und der ermittelten Position.

Der maximale durchschnittliche Fehler beträgt hier 1,44 m. Betrachtet man jedoch die Durchschnittswerte der einzelnen Felder, fällt auf, dass teil-

weise größere Sprünge (bis zu 2 m) aufgetreten sind.
Um zu verdeutlichen, welche Auswirkung solche Sprünge auf die Position der virtuellen Elemente haben, stelle man sich vor, man stünde einem POI direkt gegenüber und die Informationen würden im besten Fall in der Mitte des Bildschirms visualisiert. Würde man aufgrund einer Fehlerkennung näher zum POI hin oder weiter vom POI weg positioniert werden, hätte dies zunächst keine Auswirkung auf die Position des POIs, denn die Richtung ist durch Entfernungsänderungen konstant. Sobald man jedoch seitlich falsch positioniert werden würde, würden die POI-Informationen entsprechend versetzt im Bild visualisiert werden. Bereits bei einem halben Meter wäre ein Pixelunterschied von 119 px (ca. 19% des Bildschirms der Google Glass) denkbar, bei 1 Meter 225 px(ca. 35%), bei 2 Metern wäre das POI nur noch teilweise bzw. bei 3 Metern gar nicht mehr auf dem Bildschirm sichtbar[74]. Daran wird deutlich, welche hohen Anforderungen eine Indoorortsbasierte AR Anwendung an die Positionsbestimmung stellt.
Das Hauptproblem sind jedoch nicht die Sprünge selbst, sondern vielmehr die Häufigkeit ihres Auftretens. Ein Benutzer empfindet seltene weite Sprünge weitaus weniger störend, als häufige kleinere, denn das Bild wirkt nicht fließend sondern eher stockend und wird vom Benutzer als nicht real wahrgenommen. Wurde also eine Position erkannt, sollte sich diese möglichst stabil fortsetzen, um den Verschmelzungsgrad zwischen Realität und Virtualität zu unterstützen. Um weitere Stabilität zu gewinnen sind daher sicherlich noch weitere Optimierungen notwendig.
Um große Sprünge durch Ausreißer zu reduzieren, wären verschiedene Verfahren denkbar. King et al. verwenden einen Wahrscheinlichkeitsvektor, der alle ermittelten Wahrscheinlichkeiten in der Online-Phase speichert und bei jedem Durchlauf aktualisiert. Dabei werden vor allem Wahrscheinlichkeitsdaten aus der Vergangenheit zur Berechnung der aktuellen Position berücksichtigt:

$$\pi_i' = \frac{\pi * P(M \mid p_i)}{\sum_{j=1}^{n} \pi_j * P(M \mid p_j)} \tag{8}$$

Diese Variante wurde während der Analysephase implementiert und getestet. Die empirischen Ergebnisse zeigten, dass mit Erhöhung der Stichproben in der Online-Phase unter Einbeziehung der Daten aus der Vergangenheit die Genauigkeit erhöht wurde, die Wahrscheinlichkeits-Folge einer Position jedoch bereits nach 6-8 Stichproben gegen die Wahrscheinlichkeit 1, während die Restwahrscheinlichkeiten gegen 0 konvergierten. Dies hat zur Folge, dass der Algorithmus schnell terminiert, da eine vorherige ermittelte 0-Wahrscheinlichkeit in der obigen Formel letztlich immer zu 0 führt. Eine Erkennung eines Positionswechsels ist dann nicht mehr möglich. Grundsätzlich musste festgestellt werden, dass die Warscheinlichkeitsverteilungen

[74] Ausgehend von einer Bildschirmbreite von 640 px und 75,7°, siehe Kapitel 3.3.3

zu schnell konvergieren, sodass Positionsänderungen nur verzögert sichtbar wurden. Allerdings wurde auch deutlich, dass durch Einbeziehung von Vergangenheitsdaten die Positionsbestimmung noch weiter optimiert werden konnte.
Oft gelten viele Referenzpunkte um den gesuchten Punkt herum als wahrscheinlich, sodass die gesuchte Position durch das arithmetische Mittel der k-wahrscheinlichsten Positionen ermittelt werden kann:

$$p^* = \frac{1}{k} * \sum_{i=0}^{k} p_{max(i,P(M|p_j)} \tag{9}$$

Eine weitere Ursache für Fehlerkennungen, könnte die Zelldichte sein. Die obigen Tests wurden in einem Gebäude mit drei AccessPoints durchgeführt, wobei zusätzlich die AP's der nebenstehenden Gebäuden berücksichtigt wurden. Dies führte dazu, dass ein Großteil der Signalstärken in einem sehr tiefen Bereich aufgetreten sind, da sie von AP's aufgenommen werden, die weiter entfernt waren. Aufgrund der Empfängerempfindlichkeit des Adapters ist denkbar, dass ein Großteil der AP's in den Stichproben gar nicht oder nur zum Teil vorkamen. Die Empfängerempfindlichkeit liegt beim Galaxy SII bei -90 und beschreibt die Grenze an der der Empfang eines Beacon-Frames noch wahrscheinlich ist. Anhand der Messdaten konnte man feststellen, dass 35% der Signalstärken im Intervall [-100;-91] aufgetreten sind. In diesem recht kleinen Intervall traten jedoch Signalstärken von 17 AP's (80%) auf, so dass man davon ausgehen kann, dass in der gegebenen Testumgebung immer höchstens nur die Signale von 2-3 AP's sicher empfangen werden konnten.
An dieser Stelle sei nochmal zu erwähnen, dass zur Vergleichbarkeit der Testergebnisse, die Stichproben immer in Gehrichtung aufgenommen wurden. Es fiel auf, dass sich die Ergebnisse zum Teil verbesserten, wenn die Daten in der Online-Phase aus der gleichen Himmelsrichtung aufgenommen wurden, wie in der Offline-Phase. Laut King et al.[75] entstehen diese Differenzen durch Absorptionseffekte des menschlichen Körpers, der mindestens zu 50% aus Wasser bestehe. Je nachdem, wie also das Empfangsgerät zum Benutzer und dem AP steht, würden ein Teil der Signale mit einer geringeren Empfangsfeldstärke den Empfänger erreichen. So ergeben sich schließlich auch unterschiedliche Umgebungsprofile. Grundsätzlich empfiehlt es sich also die Messdaten nicht nur in einer Richtung zu sammeln, d.h. in der Offline-Phase öfters die Position zu wechseln. King et al. stellen einen Ansatz vor, bei dem in der Offline-Phase mit jedem Signalstärkevektor zusätzlich die Orientierung gespeichert wird, die das Gerät misst. In der Online-Phase werden nur

[75] Vgl. King et al., COMPASS: A Probabilistic Indoor Positioning System Based on 802.11 and Digital Compasses, S. 2 ff.

die Messdaten berücksichtigt, die eine ähnliche Orientierung haben, wie diese der Stichprobe. Voraussetzung ist dabei ein Gerät mit Magnetfeld- und Inertialsensoren.

Zu den Vorteilen der WLAN-Fingerprinting Methode zählt sicherlich die Nutzung der bestehenden Infrastruktur der Technischen Hochschule. Dadurch könnten die Kosten gering gehalten werden. Ein wesentlicher Nachteil des WLAN-Fingerprinting ist die Notwendigkeit der Offline-Phase, die sich in einem großem Gebäude, wie das der TH, als sehr aufwendig darstellt. Es wurde deutlich, dass mindestens 30 Scandurchläufe (4 Sekunden pro Scandurchlauf) pro Position in der Offline-Phase aufgenommen werden müssen, um eine hohe Genauigkeit zu erreichen. Hier wären noch Ansätze zu verfolgen, die weniger Messdaten benötigen. Man kann jedoch davon ausgehen, dass sich die gesamte Infrastruktur immer nur geringfügig und sukzessive ändert. Zudem erhöhen die physischen Eigenschaften von Funkwellen und die Mehrwegeausbreitungen zu stark streuenden Signalstärken, die Komplexität, ein eindeutiges Umgebungsprofil zu generieren erheblich. Besonders die Instabilität, also die häufigen Positionswechsel beeinträchtigen die Qualität der ortsbasierten AR-Anwendung maßgeblich.

3.3.2.2 iBeacons Fingerprinting

Der von Apple eingeführte, proprietäre Standard iBeacon basiert auf Bluetooth Low Energy (BLE) und beschreibt kleine Bluetooth-basierte Funkbaken, die in festgelegten Zeitintervallen Funksignale aussenden. Diese enthalten in der Regel eine eindeutige UUID mit welcher die Bake identifiziert werden kann. Grundsätzlich ergeben sich hierfür zahlreiche Anwendungsmöglichkeiten. Im Rahmen dieser Arbeit dienen Bluetooth-basierte iBeacons als Alternative zu WLAN. Dabei soll ähnlich verfahren werden wie beim WLAN-Fingerprinting und dem gegenüber Optimierungsmöglichkeiten aufgezeigt werden. Die Arbeit beschränkt sich dabei ausschließlich auf die Funkbaken vom Hersteller Estimote. Diese sind mit einem eigenen Prozessor, Flashspeicher, einem Bluetooth 4.0 Chip und einer Knopfbatterie ausgestattet, um die Platine mit Strom zu versorgen. Die Knopfbatterie ist nicht austauschbar und im Plastikgehäuse fest verbaut. Sie soll laut Estimote etwa 3 Jahre halten.

Estimote liefert zu den iBeacons zudem eine eigene Programmierschnittstelle mit, um mit diesen zu kommunizieren. Zudem eine speziell für die Indoor-Positionierung entwickelte API, die jedoch nur für IOS Geräte un-

[76] o. V., Adding Real-world Context with Estimote Beacons and Stickers | Xamarin Blog

Abbildung 17: Estimote Beacon Hardware [76]

terstützt wird. Da die Google Glass jedoch auf Android basiert, kann die API vorerst nicht genutzt werden. Zudem gibt der Hersteller hier eine Genauigkeit von bis zu 4 m an, die für ein ortsbasiertes System nicht tragbar wäre. Da es sich bei iBeacon um einen Apple Standard handelt, stellt Estimote für Android grundsätzlich nur eine stark eingeschränkte API zur Verfügung, mit der sich lediglich Informationen von den Funkbaken abfragen lassen. Für das Fingerprinting-Verfahren reicht die MAC-Adresse und die empfangene Signalstärke jedoch aus. Das Fingerprinting-Verfahren unterscheidet sich bei Bluetooth nur geringfügig. Die Signalstärken der iBeacons werden im gleichen Signalstärke-Intervall $[-100; 0]$ wie bei WLAN gemessen, wobei man feststellen kann, dass bei Werkseinstellung der Baken die Signalstärke selbst bei 1 cm Entfernung nicht über -50 reicht.
Jeder Beacon besitzt mehrere Konfigurationseigenschaften, darunter ein Name, eine UUID, die Sendeleistung (Transmit Power Tx) und das Sendeintervall (Advertising Interval). Letztere beiden haben laut Hersteller Einfluss auf die Genauigkeit der Abstandserkennung.
Die Sendeleistung beschreibt die Leistung der iBeacon-Antenne, von welchem das Signal ausgeht. Hierzu schreibt der Hersteller[77]: „The more power, the longer the range.“ (Umso mehr Leistung, umso größer die Reichweite). Sendet also der iBeacon mit höherer Leistung, können Pfadverluste reduziert werden, wobei nach wie vor gilt: Umso weiter man vom iBeacon entfernt ist umso instabiler das Signal. Die Konfiguration reicht von -30 dBm bis 4 dBm. Estimote weist allerdings darauf hin, dass durch Erhöhung der Leistung der Batterieverbrauch steigt. Tabelle 3 stellt die Leistung zur maximalen Distanz und Batterieverbrauch gegenüber.

Das Sendeintervall beschreibt den Zeitabstand zwischen den Broadcast-Beacons, die die iBeacons regelmäßig senden. Der Hersteller äußert sich da-

[77] Vgl. o. V., What are Broadcasting Power, RSSI and other characteristics of beacon's signal? – Estimote Community Portal

Leistung (dBm)	Distanz (m)	Lebensdauer Batterie (Monate)
-30	1,5	38
-20	3,5	37
-16	7	37
-12	15	36
-8	30	34
-4	40	33
0	50	30
4	70	26

Tabelle 3: Angaben von Estimote zu Leistung, Distanz und Batterieverbrauch (Ausgehend von Werkseinstellung)

zu wie folgt: „ The shorter the interval, the more stable the signal“[78] (Umso kürzer das Intervall umso stabiler das Signal). Sendet zudem der iBeacon in sehr kurzen Abständen soll es möglich sein, Ausreißer, die durch Mehrwegeausbreitungen entstehen entsprechend zu kompensieren. Das Sendeintervall reicht von 2000 ms bis zu 100 ms und hat erheblichen Einfluss auf den Batterieverbrauch. Beim höchsten Sendeintervall ist die Batterie erst nach 70 Monaten verbraucht. Bereits durch Senkung des Sendeintervalls um 20 ms, hält der Akku jedoch bereits 1 Monat weniger. Es muss also ein Kompromiss zwischen Batterieverbrauch und Genauigkeit der Positionsbestimmung gefunden werden.
Insgesamt wurden zur Evaluation 3 Versuche durchgeführt. Im ersten Versuch blieb die Konfiguration der iBeacons unverändert mit Werkseinstellung -8 dBm und 450 ms. Ausgehend von der Werkseinstellung wurde im zweiten Versuch der Einfluss der Sendeleistung auf die Positionsbestimmung untersucht. In dem letzten Versuch, wurde wiederum die Auswirkung des Sendeintervalls getestet. Es wurden also immer jeweils nur die Parameter Sendeintervall und Sendeleistung einzeln verändert um die beiden Testreihen voneinander unabhängig zu halten.
Die Untersuchungen wurden mit 6 iBeacons durchgeführt. Diese wurden auf eine große Fläche in möglichst gleich großen Abständen verteilt, wobei in allen Versuchen die Beacons jeweils mit hoher Zelldichte relativ geringer Zelldichte (max. 7 m) getestet wurden. Um einen direkten Vergleich zum WLAN-Fingerprinting herstellen zu können, wurde die gleiche Testumgebung gewählt und in der Offline-Phase die Stichproben an den gleichen Messpunkten und in Laufrichtung entnommen. Zudem wurden wieder 30 Scandurchläufe durchgeführt. Die Dauer des Scandurchlaufs musste fest auf 4 Sekunden konfiguriert werden, da ansonsten auch damit eine Vergleichbarkeit zum WLAN-Fingerprinting nicht gegeben wäre.

[78] Vgl. o. V., What are Broadcasting Power, RSSI and other characteristics of beacon's signal? – Estimote Community Portal

Pos	**T1**	**T2**	**T3**	**T4**	∅
1	1,20	1,20	1,20	0,40	1,00
2	0,20	0,20	0,40	0,40	0,30
3	0,80	0,80	0,80	0,40	0,70
4	1,00	1,40	1,20	1,00	1,15
5	0,60	1,00	0,00	0,20	0,45
6	1,20	1,40	0,80	1,80	1,30
7	1,80	0,80	0,80	1,60	1,25
8	1,80	1,40	1,00	1,20	1,35
9	1,60	1,20	1,80	0,80	1,35
10	0,60	0,40	0,60	0,60	0,55
11	0,80	0,80	0,00	1,20	0,70
12	1,40	0,40	0,80	2,60	1,30

Tabelle 4: Testergebnisse Werkseinstellung (-8 dBm, 450 ms) - Fehler in m

(a)

Pos	**T1**	**T2**	**T3**	**T4**	∅
1	1,60	1,20	1,40	0,80	1,25
2	0,60	1,40	2,60	0,20	1,20
3	0,60	0,80	0,80	0,00	0,55
4	0,40	1,20	1,60	1,40	1,15
5	0,60	1,00	1,40	1,80	1,20
6	1,00	0,80	0,00	1,20	0,75
7	0,00	1,00	0,40	3,00	1,10
8	1,20	1,00	0,20	0,80	0,80
9	0,40	0,60	0,60	1,60	0,80
10	0,80	0,40	0,40	0,60	0,55
11	1,60	0,80	0,60	0,00	0,75
12	0,40	0,60	0,80	2,00	0,95

(b)

Pos	**T1**	**T2**	**T3**	**T4**	∅
1	0,40	1,80	0,60	0,80	0,90
2	0,20	0,40	0,80	0,40	0,45
3	0,40	0,80	0,40	0,20	0,45
4	0,20	1,60	0,60	1,60	1,00
5	0,40	1,20	1,20	0,80	0,90
6	0,00	0,20	0,00	0,00	0,05
7	0,60	0,00	0,00	0,00	0,15
8	0,60	0,20	0,80	0,20	0,45
9	0,00	0,60	0,00	0,20	0,20
10	0,80	0,80	0,80	1,00	0,85
11	0,40	0,80	1,00	0,20	0,60
12	0,60	1,20	1,20	1,40	1,10

Tabelle 5: Testergebnisse Erhöhung Sendeleistung (4 dBm, 450 ms) (a) und Testergebnisse Verringerung Sendeintervall (-8 dBm, 220 ms) (b) - Fehler in m

In den Ergebnissen des ersten Versuchs (Tabelle 4) kann man feststellen, dass der maximale durchschnittliche Fehler von 1,35 m tendenziell besser ausfällt, als bei dem WLAN-Fingerprinting. Die höhere Genauigkeit könnte auf die Anzahl der Beacons zurückzuführen sein, die mehr Testdaten als die drei AP's im Haus liefern. Trotz allem, ist die Verbesserung gegenüber des WLAN-Fingerprintings nur minimal. Vergleicht man die Verteilung der durchschnittlichen Fehler aus den beiden Tabellen, stellt man fest, dass beim Bluetooth-Fingerprinting sogar häufiger durchschnittliche Fehler über einem Meter entstehen als beim WLAN-Fingerprinting.
Eine Ursache könnte die geringe Sendeleistung der Funkbaken sein. Während die Sendeleistung von WLAN bei etwa 13 bis 18 dBm liegt, wurden die Funkbaken auf -8 dBm konfiguriert. So erkennt man in Tabelle 5b, dass durch Erhöhung der Sendeleistung (auf 4 dBm) bessere Ergebnisse erzielt

wurden, auch wenn nur geringfügig. Der maximale durchschnittliche Fehler liegt bei 1,25 m. Allerdings wird aufgrund dieser Konfiguration ca. 20% der Akkulaufzeit reduziert.
Zuletzt kann man erkennen, dass durch Verringerung des Sendeintervalls, bei gleichbleibender Sendeleistung von -8 dBm die besten Ergebnisse erzielt werden konnten (max. durchschnittlicher Fehler 1,1). Dies lässt sich vor allem auf die größere Stichprobenmenge zurückführen, da in der Offline-Phase innerhalb eines 4 Sekunden-Scandurchlaufs deutlich mehr Broadcast-Beacons empfangen wurden (da sie öfters gesendet werden) und dadurch die Wahrscheinlichkeit für Ausreißer in der Häufigkeitsverteilung weiter reduziert wurde. Zum Vergleich: Beim Testdurchgang mit Werkseinstellung wurden insgesamt 1803 Strichproben entnommen, während durch Reduzierung des Sendeintervalls 2614 Stichproben entnommen wurden. Letztlich geht jedoch die leichte Verbesserung der Genauigkeit mit einem hohen Batterieverbrauch (nur mehrere Monate) aufgrund des häufigeren Aussenden der Broadcast-Beacons einher.

Insgesamt konnte gezeigt werden, dass mithilfe von Bluetooth eine hohe Genauigkeit bei der Positionserkennung im Indoor-Bereich erreicht werden kann, teilweise sogar eine höhere als beim WLAN-Fingerprinting, wenn auch nur geringfügig. Zudem wurde deutlich, dass eine Erhöhung der Sendeleistung und Verringerung des Sendeintervalls zur weiteren Optimierung beitragen kann. Allgemein ist der wesentliche Nachteil der Estimote iBeacons die im Gehäuse verbaute Batterie, die bei normalen Verbrauch nach 3 Jahren verbraucht ist. Spätestens nach 3 Jahren, müsste also ein Wechsel der gesamten Infrastruktur erfolgen und erneut die Offline-Phase des Fingerprintings durchgeführt werden. Dazu kommt, dass die einmaligen Anschaffungskosten für den Aufbau der gesamten Infrastruktur des TH-Gebäudes bereits sehr hoch sind. Der Aufwand der hierfür notwendig wäre, wird dem Nutzen, durch die Verwendung der iBeacons, daher keinesfalls gerecht. Vielmehr sollte in Maßnahmen investiert werden, die das WLAN-Fingerprinting mit der bestehenden WLAN-Infrastruktur optimieren.

3.3.2.3 LLA Marker

Sollte das Gebäude keine oder keine ausreichend dichte WLAN oder iBeacon-Infrastruktur besitzen, bzw. diese aufgrund eines technischen Defekts nicht zur Verfügung stehen, sollte es dennoch möglich sein, eine Positionsbestimmung innerhalb des Gebäudes zu ermöglichen. Eine Möglichkeit besteht darin sogenannte LLA Marker[79](Latitude, Longitude, Altitude) zu verwenden. Der Begriff des LLA Markers wurde erstmals von Metaio, einem auf

[79]Vgl. metaio, Indoor Location Based Channels

AR spezialisierten Unternehmen eingeführt. Man versteht darunter spezielle Marker, die geodätische Daten kodieren und somit erlauben, die genaue Position eines Gerätes zu erkennen.

Das Prinzip der Positionsbestimmung mit LLA Markern ist wie folgt: Innerhalb des Gebäudes werden LLA Marker an verschiedenen Positionen angebracht z.B. an Wänden oder Türen, am Boden usw. Jeder Marker codiert seine eindeutige Position in Form von Koordinaten. Metaio verwendet hierbei den Breiten- und Längengrad als geographische Koordinaten, daher die Bezeichnung LLA. Ein Markendetektor im Gerät erkennt den LLA-Marker über ein optisches Bilderkennungsverfahren und erhält damit seine Position. Eine Kamera ist also zwingend notwendig. Abbildung 18 veranschaulicht einen solchen LLA Marker. Das von Metaio entwickelte Framework Junaio verwendet einen 9x9 Raster. Jedes Feld kann dabei den binären Wert 1 oder 0 kodieren, wodurch sich 2^{81} Möglichkeiten ergeben. Während ein Teil der Bits für die Parität verwendet werden, reichen die restlichen Bits zur Kodierung des Breiten- und Längengrades.

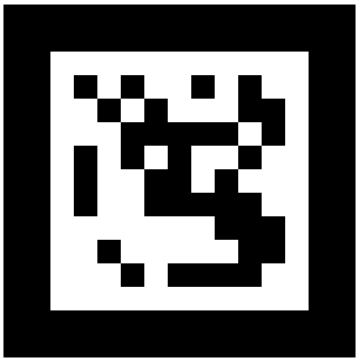

Abbildung 18: LLAMarker

Wie in den Grundlagen behandelt, werden Marker über ein optisches Verfahren eingelesen. Zur optimalen Erkennung des Markers spielen mehrere Faktoren eine Rolle. Hierzu zählt die Auflösung der Kamera, die Distanz zum Marker, die Größe des Rasters und die Größe des Markers selbst. Soll also ein Marker aus einer größeren Distanz erkennbar sein, sollte bei einer geringen Auflösung der Kamera immer ein größeres Raster bzw. ein größerer Marker verwendet werden. Dies hat zur Folge, dass u. U. nicht genügend Daten kodiert werden können. Es muss also ein Kompromiss zwischen dem maximalen Leseabstand eines Markers bzw. dessen Lesbarkeit und dessen Datenkapazität gefunden werden. Die Marker des Frameworks Junaio haben den Nachteil, dass sie ein recht kleines Raster verwenden, sodass sich die maximale Lesedistanz je nach Kamera auf etwa 30 cm beschränkt. Be-

trachtet man ein solches Szenario aus rein ergonomischen Gesichtspunkten, wäre ein Smartphone bei einer solchen Distanz noch aufgrund der hinzukommenden Armlänge denkbar. Stellt man sich das Szenario mit einer Datenbrille, wie die Google Glass vor, wird deutlich, dass ein größerer Leseabstand erforderlich wäre.
Das C++ basierende Open-Source Framework AR-Framework ArUco verwendet standardmäßig ein größeres Marker-Raster und eignet sich für die Arbeit daher nicht nur durch seine Schlankheit und Integrationsfähigkeit.Dieses verwendet standardmäßig ein 5x5-Raster, wodurch 25 Bit kodiert werden können. Wobei 10 Bit für Daten und 15 Bit für die Fehlerbehandlung vorgesehen sind. Auf die Kodierung der Position in x,y-Koordinaten muss aufgrund der geringeren Datenkapazität also verzichtet werden. Stattdessen wird jedem Marker eine eindeutige Identifikationsnummer (ID) zugewiesen. Innerhalb der Datenbank wird jede ID einer festen Position zugeordnet. Auf diese Weise können $2^{10} = 1024$ Positionen kodiert werden.
Die Funktionsweise der Markererkennung soll im Folgenden grob umrissen werden[80]:

1. Zunächst wird das von der Kamera aufgenommene Bild in Graustufen konvertiert. Da ein Marker nur aus einem schwarz-weiß-Raster besteht, sind Farbinformationen überflüssig. Eine Vielzahl von Bilderkennungsalgorithmen arbeiten in der Regel mit Graustufen, da diese die Anzahl an Farb-Informationen pro Pixel (0-255) deutlich reduzieren und sich somit positiv auf die Performance auswirken.

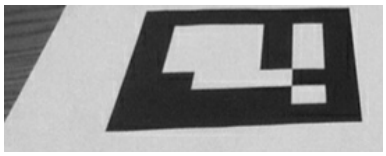

Abbildung 19: Bild in Graustufen umgewandelt [81]

2. Im zweiten Schritt wird das Bild binarisiert, d.h. jedem Pixel statt 0-255 einen Wert zwischen 0 oder 1 zugeordnet. Auf diese Weise können später Umrisse und Formen besser klassifiziert werden. Zudem werden die Marker besser vom Hintergrund abgehoben. Je nach Verteilung der Grauwerte, die auch vom natürlichen Licht beeinflusst werden, stehen für die Umwandlung verschiedene Schwellwertmethoden zur Verfügung.

[80] Vgl. Baggio et al., *Mastering OpenCV with Practical Computer Vision Projects* -, S.69
[81] A. a. O.
[82] A. a. O.

Abbildung 20: Binarisiertes Bild[82]

3. Durch die Binarisierung im zweiten Schritt lassen sich Schwarz-Weiß-Übergange als Konturen erkennen, wodurch im nächsten Schritt mit einem speziellen Verfahren zur Konturfindung alle Polygone im Bild erkannt werden können. Hier werden bereits Schwellwerte definiert um zu kleine Konturen auszuschließen. Nachdem die Polygone vom Algorithmus zurückgegeben wurden, können solche eliminiert werden, die weniger oder mehr als 4 Punkte besitzen.

Abbildung 21: Bild nach Konturfindung[83]

4. Zum Schluss wird ein Ausschnitt, welcher durch das Polygon definiert wird, aus dem Bild entnommen und per Koordinatentransformation in ein zweidimensionales und immer gleich großes Bild überführt. Auf diese Weise lässt sich ein gleichbleibendes Gitter aufbauen, in welchem jedes Feld entweder mit einer überwiegenden Anzahl an schwarzen oder weißen Pixel enthält. Das Gitter besteht aus 5 x 5 Bit langen

Abbildung 22: Bild nach Koordinatentransformation[84]

Hamming Codes[85], wobei nur das 2. und 4. Bit die eigentlichen Daten speichert, während die restlichen Bits Paritätsbits darstellen. Ferner wird das 1. Paritätsbit invertiert, um einen vollständig schwarzen Kasten und damit die Anzahl falscher Negative bei der Markererkennung zu minimieren. Die hohe Anzahl an Paritätsbits soll zudem gewährleisten, stets die Orientierung des Markers ermitteln und den Marker

[83] Baggio et al., *Mastering OpenCV with Practical Computer Vision Projects* -
[84] A. a. O.
[85] Vgl. o. V., ARUCO How to create markers | iplimage

entsprechend transformieren zu können.

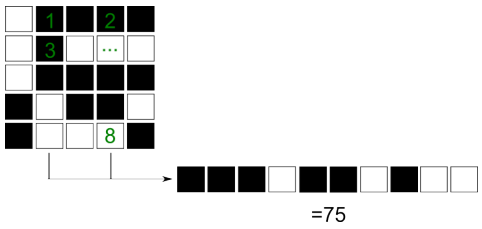

Abbildung 23: ArUco Marker[86]

Da jeder LLA Marker immer genau eine Position kodiert, müsste theoretisch an jeder Position ein Marker angebracht werden. Da dies zu aufwendig wäre, werden die LLA Marker nur an einigen Stellen angebracht, an welchen dann eine Positionsbestimmung möglich ist. Da man grundsätzlich jedoch einen umfangreichen Überblick aller POIs in der Umgebung erhalten möchte, sollten sich diese möglichst zentral im Raum befinden, sodass man diese in der Regel nur auf dem Boden platzieren kann.

Zu den Vorteilen der LLA Markern zählt sicherlich die Einfachheit und der geringe Aufwand, die Marker herzustellen und an den jeweiligen Stellen zu platzieren. Zudem weißt die Positionsbestimmung eine hohe Stabilität auf, da bis zum Scan des nächsten Markers kein Positionswechsel stattfindet. Dies geht jedoch mit dem Nachteil einher, dass aufgrund des Aufwands nicht jede Position im Koordinatensystem abgebildet werden kann und die Position nur solange korrekt ist, bis der Benutzer von der Position abweicht.

3.3.3 Einblendung von Point of Interests

In den vorherigen Kapiteln wurde ein Koordinatensystem als Umgebungsmodell aufgestellt, welches jede Position in Form von x,y-Koordinaten beschreibt. Ferner wurden Verfahren untersucht, um die Position des Benutzers innerhalb des Koordinatensystems zu bestimmen. Zum jetzigen Stand könnten damit also theoretisch alle naheliegenden POIs, relativ zur aktuellen Position ermittelt werden. Damit können die Fragen „Wo bin ich?“ und „was ist in meiner Nähe?“ beantwortet werden. Im letzten Schritt geht es darum diese Informationen im Blickfeld des Benutzers zu visualisieren („Was sehe ich?“) und damit erweiterte Realität zu ermöglichen.
Eines der Ziele dieser Arbeit ist die Realisierung von Optical-See-Through AR mit der Google Glass. Betrachtet man erneut die Definition von Optical-See-Through aus Kapitel 2.1.2.1 ergeben sich daraus einige Grundvoraussetzungen. Zum einen wird ein Anzeigegeräte mit einem optischen Kom-

[86] o. V., ARUCO How to create markers | iplimage

binierer benötigt, um die reale Welt mit dem virtuellen Bild zusammenzuführen. Das Prisma der Google Glass ist ein semi-transparentes Display, welches ein Teil des natürlichen Lichtes mit einem Teil des künstlich erzeugten Lichtes aus einem Projektor zusammenführt und damit die Funktion eines optischen Kombinierers erfüllt. Auf diese Weise ist das Hauptmerkmal eines Optical-See-Through-Systems gegeben, denn die real wahrgenommene Welt kann um Virtualität erweitert werden. Allerdings besitzt die Google Glass in diesem Kontext einige Einschränkungen:
Zunächst befindet sich das virtuelle Bild im oberen rechten Sichtbereich und wird erst scharf, wenn der Benutzer den Blick zum Bild richtet. Das Registrieren, also das Einfügen des virtuellen Bildes in das real wahrgenommene Bild wird dadurch sichtlich erschwert. Dies stellt vor allem bei objektbasierten AR Applikationen ein Problem dar, da die virtuellen Inhalte in Abhängigkeit eines realen Objektes präzise positioniert werden müssen. Dies ist einer der wahrscheinlichsten Gründe, warum bisher für die Google Glass nur Video-See-Through AR Apps entwickelt wurden, die das reale Bild mit der Kamera aufnehmen, dieses mit virtuellen Objekten kombinieren und schließlich im Display ausgeben. Bei einer ortsbasierten AR-Applikationen hingegen, spielt eine präzise Registrierung eher eine untergeordnete Rolle. Vielmehr geht es bei ortsbasierter AR darum, ortsabhängige Informationen zu visualisieren. Dies setzt jedoch voraus, dass ein Umgebungsmodell existiert und eine Positionsbestimmung möglich ist.
Die zweite Einschränkung ergibt durch das semi-transparente Display. Da das Prisma der Google Glass nur einen Teil des Lichtes für das virtuelle Bild in die Retina des Betrachters projiziert, wirkt das Bild oft blass. Zudem geht das von der Brille erzeugte Bild nicht mit der Realität ineinander über, da es immer einem 640x360 großen Bereich, oft mit schwarzem Hintergrund abgebildet wird. Aufgrund dieser Tatsache, wird auch eine direkte Verschmelzung des realen und virtuellen Bildes erschwert. Da die Verschmelzung einen Qualitätsfaktor eines AR-Systems darstellt, mussten zunächst einige Optimierungen vorgenommen werden.
Grundsätzlich wird die Transparenz des Bildes durch drei Faktoren beeinflusst: die Lichtintensität des natürlichen Lichtes, die des Projektors und der Farbwert des Pixels. Auf die letzten beiden, hat man grundsätzlich Einfluss. Schwarze Pixel erscheinen besonders transparent, während weiße Pixel das natürliche Licht überblenden. Aus diesem Grund wurde für die AR-Applikation ein schwarzer Hintergrund gewählt, um den Hintergrund möglichst Transparent zu halten. Zudem kann das vom Projektor, virtuell erzeugte Licht reduziert werden. Dies geht jedoch mit dem Nachteil einher, dass das virtuelle Bild in der Gesamtheit noch blasser erscheint. Es muss also ein Kompromiss zwischen der Sichtbarkeit des Bildes und der Grad der Verschmelzung gefunden werden.

Eine weiterer Grundbaustein eines Optical-See-Through Systems ist die Tracking-Komponente. Wie in den Grundlagen beschrieben, dient sie zur Erfassung der Pose, um damit Kontext-abhängige Informationen zu visualisieren. Im Rahmen dieser Arbeit beschränkt sich dabei der Kontext auf Informationen, die einen Bezug zur aktuellen Position innerhalb eines Gebäudes und zum aktuellen Blickfeld haben. Richtet etwa der Benutzer den Blick auf einen Point of Interest, also einen bestimmten Raum o.ä. im Flur des Gebäudes, sollen ihm bestenfalls nur Informationen zu diesem visualisiert werden. Eine Grundvoraussetzung wurde hierfür bereits durch ein Umgebungsmodell und die Positionsbestimmung geschaffen, mit welcher die x- und y-Position und damit bereits zwei Freiheitsgrade gemessen werden können.
Um jedoch vollständig die reale Umgebung zu erfassen, müssen theoretisch alle 6 Freiheitsgrade bekannt sein. Da in dieser Arbeit von einem zweidimensionalen Koordinatensystem ausgegangen wird, werden also noch die Roll-Nick-Gier-Winkel benötigt, um die Pose zu ermitteln. Der Roll-Winkel beschreibt dabei die Neigung nach links und rechts, Nick-Winkel nach oben und unten und Gier-Winkel die Kopf- bzw. Körperdrehung, also die Blickrichtung.

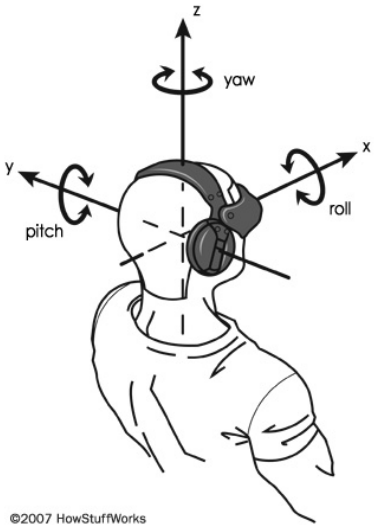

Abbildung 24: Roll-Nick-Gier[87]

Auch hierfür stellt die Google Glass bereits die notwendigen technischen Voraussetzungen bereit. Mithilfe des eingebauten Accelerometers kann die lineare Beschleunigung (m/s^2) Richtung x,y,z-Achse gemessen und damit bereits die Orientierung aus den Winkelunterschieden des Geräts bestimmt

[87] o. V., Virtual Reality Tracking Systems - HowStuffWorks

werden. Ferner besitzt die Google Glass ein Gyroskop, das zur Messung der Winkeländerungsgeschwindigkeit dient. Über die Änderung des Winkels um eine Achse in der Zeit, kann durch Integration nach Zeit der Änderungswinkel um alle Achsen bestimmt werden. Durch die Einbeziehung des Gyroskops kann die Genauigkeit deutlich erhöht werden. Das Accelerometer und Gyroskop befinden sich im vorderen Sockel der Google Glass und liefern die Messwerte in einem eigenen Koordinatensystem zurück, in welchem die Y-Achse nach oben, die X-Achse nach rechts und Z-Achse in Richtung des Betrachters zeigt (siehe Abb. 25). Allerdings können hieraus noch nicht die Roll-Nick-Gier Winkel abgeleitet werden, da beide Sensoren zum driften neigen, wenn sich Fehler akkumulieren, sodass zusätzlich ein Referenzsystem benötigt wird. Erst durch Nutzung eines Magnetometers, welcher die Ausrichtung zum magnetischen Norden (Gier) bestimmt, stehen alle Informationen zur Verfügung, um die Rotation um alle Achsen und damit die Roll-Nick-Gier-Winkel zu ermitteln. Die Android API stellt hierfür bereits die notwendigen Funktionen bereit, um die Fusion aller Sensoren zu ermöglichen.

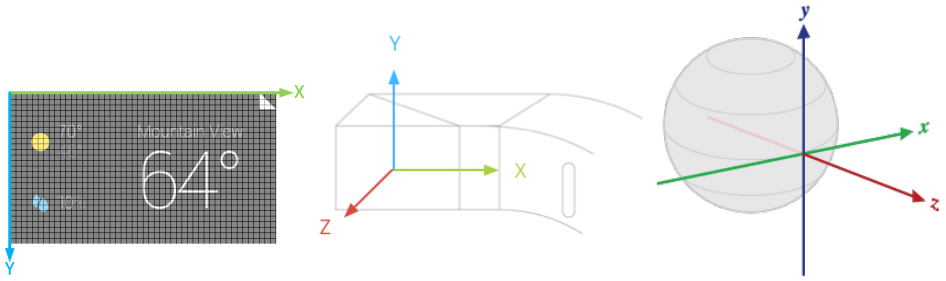

Abbildung 25: Unterschiedliche Koordinatensysteme[88]

Zurückgegeben wird schließlich ein Rotationsvektor $\vec{r} = (\alpha, \beta, \gamma)$, der die Rotation des Gerätekoordinatensystems um das Weltkoordinatensystem beschreibt. In Abbildung 25 kann man erkennen, dass jedoch das Koordinatensystem der Google Glass gegenüber dem Weltkoordinatensystem um 90 Grad gekippt ist. Würde man die Google Glass mit dem Sockel auf einen flachen Tisch stellen, würden die Gier-Werte grundsätzlich korrekt zurückgegeben werden, da in diesem Falle die Z-Achse als Gier-Achse übereinstimmen würde. Da jedoch beim Tragen der Brille die Y-Achse die Gier-Achse ist, muss diese auf die Z-Achse abgebildet werden. Die Android API stellt auch hierfür Methoden bereit um eine solche Koordinatentransformationen durchführen zu können.
Nachdem die drei weiteren Freiheitsgrade ermittelt wurden, müssen die Winkelinformationen in Bildschirmkoordinaten abgebildet werden. Abbildung 25 veranschaulicht das Display-Koordinatensystem. Hierbei stellt die

[88] o. V., Locations and Sensors | Glass | Google Developers, o. V., SensorManager | Android Developers

X-Achse die horizontale und Y die vertikale Achse dar, wobei der Ursprung oben links liegt.
Die horizontale Positionierung der virtuellen Inhalte erfolgt über den Gier-Winkel α, da eine Drehung des Kopfes nach links und rechts eine Bewegung der Inhalte auf der Horizontalen bewirken soll. Damit eine horizontale Positionierung in Abhängigkeit des Gier-Winkels ermöglicht werden kann, muss im ersten Schritt die Richtung zum umliegenden POI, auch Peilung genannt, ermittelt werden. Die Peilung beschreibt einen Winkel zwischen der Richtung eines gepeilten Objektes und der gemessenen Himmelsrichtung. Im Falle der Google Glass ist die Blickrichtung mit der Himmelsrichtung identisch. Nur auf diese Weise lässt sich aus der Blickrichtung ermitteln, welche POIs sich aktuell im Sichtfeld befinden. Die Peilung φ von der aktuellen Position (x_p, y_p) zu einem POI (x_{poi_i}, y_{poi_i}) errechnet sich wie folgt:

$$\varphi_{\overrightarrow{ppoi_i}} = ((90 - \frac{180}{\pi} * \text{atan2}\,(y_p - y_{poi_i}, x_{poi_i} - x_p)) - \delta) \mod 360 \quad (10)$$

Die atan2-Funktion berechnet den Winkel zwischen dem Richtungsvektor, der geographisch gesehen nach Norden zeigt und dem Vektor $\overrightarrow{ppoi_i}$. Die y-Werte der aktuellen Position und die des POIs wurden in der Formel vertauscht, da der Ursprung des erzeugten Koordinatensystems oben links statt oben rechts beginnt. Der Wert wird anschließend in Grad umgerechnet. Da das Polarkoordinatensystem seinen Ursprung bei der X-Achse des kartesischen Koordinatensystems hat, muss der von atan2 errechnete Winkel nochmals um 90 Grad in den Ursprung des kartesischen Koordinatensystems verschoben werden. δ bezeichnet die Abweichung zwischen dem magnetischen Norden und dem relativen Norden im erzeugten zweidimensionalen Gitternetz. Damit wird gewährleistet, dass das erzeugte Umgebungsmodell immer mit dem Weltkoordinatensystem übereinstimmt, auch wenn ein Gebäudeplan nicht gegen Norden zeigt. Ferner muss das horizontale Sichtfeld (FOV) im Verhältnis zum Display eingeschränkt werden, damit nur virtuelle Objekte visualisiert werden, die sich innerhalb des Sichtfelds befinden. Das Sichtfeld ist also für die Anzeige von Kontext-abhängigen Informationen entscheidend. Hierbei wurde das Sichtfeld der Kamera gewählt, welches bei der Google Glass etwa 75,7° beträgt. Schließlich errechnet sich die horizontale Bildschirmkoordinate (in Pixel) wie folgt:

$$x_{poi_i} = \frac{Bildschirmbreite}{FOV_h} * (\alpha - \varphi_{\overrightarrow{ppoi_i}}) + \frac{Bildschirmbreite}{2} \quad (11)$$

Zunächst wird die Anzahl der Pixel pro Grad aus der Bildschirmbreite, die bei der Google Glass 640 Pixel beträgt, und dem horizontalen FOV berechnet. Dieser Wert wird mit der Differenz aus dem ermittelten Gierwinkel des Sensors und der Peilung zum jeweiligen POI multipliziert. Als Bezugs-

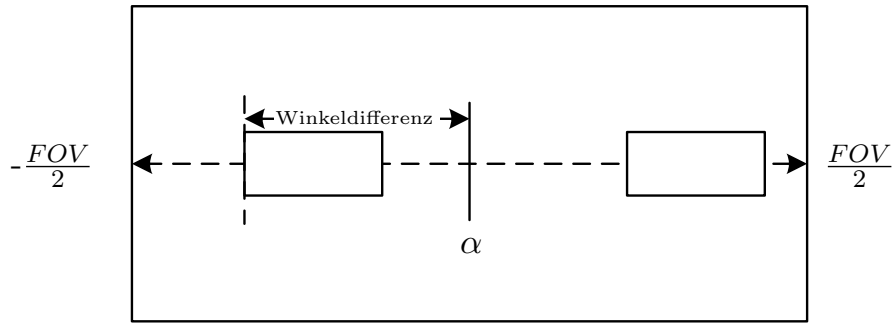

Abbildung 26: Horizontale Positionierung in Abhängigkeit der Winkeldifferenz vom Gierwinkel zur POI-Peilung

punkt wird die Position des Zentrums hinzu addiert. Die Bildschirmbreite und FOV ist konstant, daher entscheidet die Winkeldifferenz vom Gierwinkel zur Peilung des POIs, wie das virtuelle Objekt positioniert wird.
Der Nick-Winkel β bestimmt wiederum die vertikale Positionierung, denn durch Beugung des Kopfes nach vorne oder hinten sollen die virtuellen Objekte vertikal ihre Position verändern. Zur Berechnung der vertikalen Position der virtuellen Elemente wird wiederum immer vom Zentrum ausgegangen, wenn der Nick-Winkel 0° beträgt. Das vertikale Sichtfeld FOV_v der Google Glass Kamera beträgt 58.3°.

$$y_{poi_i} = \frac{Bildschirmbreite}{FOV_v} * \beta + \frac{Bildschirmbreite}{2} \tag{12}$$

Mithilfe des Roll-Winkels γ wird schließlich die Rotation ζ der virtuellen Objekte bestimmt. Hier gilt ohne Umrechnung $\zeta = \gamma$.

Zum jetzigen Konzeptstand würden alle POIs visualisiert werden, die sich im aktuellen Sichtfeld befinden und damit u. U. auch POIs, die in der realen Welt von Hindernissen verdeckt werden. Beispielsweise würden beim Blick auf einen Raum in einem Flur auch Informationen zu anderen Räumen visualisiert werden, wenn sich diese etwa im Parallelflur befinden. Um dies zu vermeiden müssen zusätzliche Informationen zum Umgebungsmodell herangezogen werden. Daher wird das in Kapitel 3.3.1 Umgebungsmodell zusätzlich um Hindernisinformationen erweitert. Jede Position kann dann ein Hindernis, wie z.B. eine Wand sein. Um zu bestimmen, ob ein POI sich hinter oder vor einem Hindernis befindet, wird aus der aktuellen Position und der Position des ausgewählten POIs eine Gerade

$$\vec{x} = \vec{p} + s\,\vec{u} \tag{13}$$

ermittelt. Da das definierte Koordinatensystems nur diskrete Werte erlaubt, können die n Punkte zwischen den beiden Punkten innerhalb der Gerade

durchlaufen werden. Handelt es sich bei einem der Punkte um ein Hindernis, wird das POI nicht als virtuelles Objekt visualisiert.
Wie erwähnt, werden die virtuellen Objekte einem zweidimensionalen Koordinatensystem abgebildet. Also wird die vom Auge wahrgenommene, dreidimensionale Realität durch zweidimensionale Inhalte erweitert. Um dennoch die virtuellen Elemente korrekt in den dreidimensionalen Raum einzufügen und damit eine Tiefenwirkung zu erzeugen, werden die virtuellen Objekte je nach Distanz zum POI entsprechend skaliert. Die Distanz zwischen der aktuellen Position und dem POI ergibt sich aus dem euklidischem Abstand zwischen zwei Punkten:

$$D_i = \sqrt{(x_p - x_{poi_i})^2 + (y_p - y_{poi_i})^2} \tag{14}$$

Je nach Distanz werden die virtuellen Objekte also größer bzw. kleiner abgebildet, sodass vorrangig die nächsten POIs im Fokus stehen. Ferner werden ab einem bestimmten Schwellwert die Objekte nicht mehr visualisiert, damit der Benutzer nicht an Informationsüberfluss leidet.

3.4 Navigation

Damit der Benutzer nicht nur Informationen zu seiner Umgebung erhält, sondern auch innerhalb des Gebäudes zu Räumen navigieren kann, wird zusätzlich eine Indoor-Navigation benötigt. Dabei soll sich der Benutzer anhand von virtuellen Inhalten von einer Position zur anderen leiten lassen und immer der kürzeste Pfad gewählt werden. Die notwendigen Voraussetzungen der Indoor-Navigation wurden bereits mit dem Konzept für das AR-System geschaffen. Mit den Verfahren zur Positionsbestimmung sollte man in der Lage sein, die ungefähre Position des Benutzers innerhalb des Gebäudes zu ermitteln. Im zweiten Schritt muss ein optimaler Weg zum Ziel ermittelt werden. Um dies zu ermöglichen, muss das bestehende Umgebungsmodell nochmals erweitert werden.
Das bestehende Umgebungsmodell ist ein Regular-Grid bestehend aus $n*m$ Feldern, die entweder Positionsdaten, POIs und Hindernisse speichern. Damit ein Pathfinding-Algorithmus in der Lage ist, das kürzeste-Pfad-Problem zu lösen, muss das bestehende Raster in einen Graph transformiert werden, in welchem jedes Feld als Knoten und der Übergang von einem Feld zum anderen als Kante betrachtet wird. In dem erzeugten Raster, bestehen an jedem Feld vier Traversierungsmöglichkeiten. Der Aufwand von einem Feld zum anderen zu gelangen beträgt immer 1. Werden zusätzlich die Diagonalen berücksichtigt, würden sich weitere vier Kanten mit dem Aufwand $\sqrt{1^2 + 1^2} = 1.41$ ergeben. Damit Hindernisse bei der Ermittlung des Pfades nicht traversiert werden, werden nur die Felder bzw. Knoten miteinander

verknüpft, wenn diese keine Hindernisse darstellen. Mithilfe des Dijkstra-Algorithmus, wird anschließend der kürzeste Pfad von einem Knoten zum anderen ermittelt.
Abbildung 27 veranschaulicht, wie der Dijkstra-Algorithmus an der Karte verlaufen würde. Der gesamte Pfad besteht hier aus 10 zu traversierenden Knoten. Würde nun der Benutzer den Pfad entlanggehen, wären insgesamt 3 Aktionen notwendig: rechts abbiegen, links abbiegen, rechts abbiegen. Dies stellt sich problematisch dar, wenn sich die Richtungsänderungen auf sehr engem Raum, also in kurzen Zeitabständen ergeben. Man stellt also fest, dass sich diese Wegführung relativ aufwendig und kontraintuitiv darstellt.

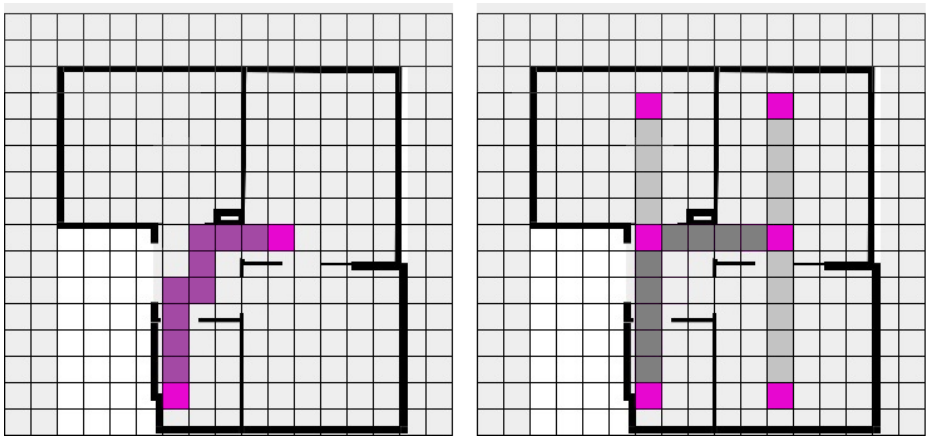

Abbildung 27: Links: Dijkstra Algorithmus mit allen Feldern. Rechts: Dijkstra Algorithmus mit Waypoints

Berücksichtigt man zudem die verwendeten Methoden zur Positionsbestimmung, wird deutlich, dass diese Art der Wegführung in Kombination nicht sinnvoll ist. Um den aktuellen Knoten im Graph zu bestimmen, und damit den Dijkstra-Algorithmus jedes mal neu zu initialisieren ist eine verlässliche und kontinuierliche Positionsbestimmung unabdingbar. Geht man jedoch davon aus, dass die LLA Marker zur Positionsbestimmung genutzt werden, müsste an jeder erdenklichen Position ein Marker positioniert werden. Würde wiederum die Positionsbestimmung über Fingerprinting erfolgen, müsste diese so exakt sein und so kurze Antwortzeiten liefern, dass der Dijkstra-Algorithmus erneut initialisiert werden kann. Da beide Fälle nicht realisierbar erscheinen, wurden stattdessen Versuche unternommen, die Anzahl der Knoten zu reduzieren.
Das Konzept der Arbeit sieht daher vor, nicht alle freien Felder bei der Wegfindung zu berücksichtigen, sondern nur solche, die einen Richtungswechsel veranlassen. In erster Linie wird hierfür ein Wegenetz konstruiert, welches die wichtigsten Laufwege berücksichtigt. Ein solches Wegenetz wird von mehreren sogenannten Waypoints aufgespannt. Waypoints stellen Punkte

innerhalb des Rasters dar, welche immer eine Aktion des Benutzers erfordern, wie bspw. das Einscannen eines neuen LLA Markers, oder der Richtungswechsel. Abbildung 27 veranschaulicht beispielhaft ein solches Wegenetz. Die hellgrauen Felder visualisieren das Wegenetz, die violetten Felder stellen die Waypoints dar. Die dunkelgrauen Felder veranschaulichen den Weg, den der Benutzer gehen würde. Zwar wird in der Theorie nie der kürzeste Weg gewählt, jedoch erfordert es in der Praxis letztlich nur eine Aktion und einen Richtungswechsel, um zum gewünschten Ziel zu kommen. Überträgt man dieses Konzept auf die Gebäudestruktur des Campus Gummersbach, stellt man fest, dass besonders durch die langen Flure und wenigen Verzweigungen diese Art der Wegführung sinnvoll erscheint.

3.5 Benutzerschnittstelle und Interaktion

Mit dem Grundkonzept wurde die Basis für ein ortsbasiertes AR-System im Indoor-Bereich geschaffen. Damit der Benutzer auch mit dem System interagieren kann, ist eine Benutzerschnittstelle unabdingbar.

Die Benutzerschnittstelle sollte dabei so konzipiert sein, dass sie den Ansprüchen des Zielgeräts gerecht wird. Hierbei gilt zu berücksichtigen, dass sich die Art der Mensch-Computer-Interaktion bei der Google Glass nochmals erheblich von anderen Zielgeräten unterscheidet. Da der Benutzer stets die Datenbrille im Blickfeld hat und diese daher maßgeblich Anteil daran hat, wie der Benutzer mit der realen Welt interagiert, werden durchaus höhere Anforderungen an die Bedienbarkeit gestellt. Aus dieser Herausforderung heraus entstanden die von Google entwickelten, fünf elementaren Design-Richtlinien, welche grundlegend die Nutzererfahrung beeinflussen[89]:

1. Design for Glass - Bei diesem Designprinzip geht es in erster Linie darum, die Art der Interaktion der Google Glass von anderen Geräten wie bspw. Smartphones oder Tablets klar zu unterscheiden. Informationen der Google Glass werden immer zeitnah und aktuell visualisiert. Im Vordergrund stehen sogenannte Mikrointeraktionen, bei welchen der Benutzer auf schnellste Weise an sein Ziel gelangt. Dies deckt sich wiederum mit dem Ziel eines AR-Systems, die Zeit der Informationsbeschaffung zu reduzieren.

2. Don't get in the way - Die Grundidee der Google Glass sieht vor, nur Informationen anzuzeigen, die aktuell benötigt werden und den Benutzer interessieren. Alle anderen Informationen sollten ausgeblendet werden, um den Nutzer nicht unnötig zu verwirren.

[89] Vgl. Firstenberg und Salas, *Designing and Developing for Google Glass - Thinking Differently for a New Platform*, S.78 ff.

3. Keep it relevant - Informationen sollten Kontext-abhängig visualisiert werden. D.h. die richtige Information, sollte zur richtigen Zeit am richtigen Ort verfügbar sein.

4. Avoid the unexpected - Da das Display der Google Glass für den Benutzer immer sichtbar ist, sollten unerwartete Ereignisse möglichst vermieden werden.

5. Build for people - Die Benutzerschnittstelle sollte für Menschen entwickelt sein. Die Bedienung sollte daher einfach und intuitiv sein. Bilder, natürliche Gesten oder Sprache sollen zur Kommunikation und Interaktion dienen. Komplizierte Menüs und Einstellungsmöglichkeiten sollten vermieden werden.

Wie bereits im Grundlagenteil erwähnt, besitzt die Google Glass ein Touchpad, mit welchem Wischgesten nach links/rechts und oben/unten möglich sind. Die Art der Interaktion mit dem Touchpad, lässt sich allerdings nicht mit einem klassischen Touchpad eines Smartphones vergleichen, da der Berührungspunkt auf der Displayfläche nicht bekannt ist. Überträgt man den die Problematik auf den Anwendungsfall der AR-Applikation, wird deutlich, dass die Interaktion mit den virtuellen Objekten auf diese Weise nicht realisierbar ist. Grundsätzlich ergeben sich hierdurch verschiedene Realisierungsmöglichkeiten, die bereits in Kapitel 2.1.2.4 vorgestellt wurden. Eine reale Möglichkeit, bestünde durch die Nutzung eines Tangible User Interfaces, in welchem die virtuellen Objekte bspw. mit dem Finger selektiert werden könnten. Dies setzt jedoch wieder ein Tracking-Verfahren voraus, dass die Komplexität des Systems deutlich erhöht. Für ein Eye-Tracking-System fehlt wiederum eine Kamera, die das Auge erfasst.
Im Rahmen dieser Arbeit wurde die Selektion durch Blickrichtung als vielversprechender Ansatz gesehen, da dieser mit geringem Aufwand realisierbar ist und die Bedienung für den Benutzer deutlich vereinfacht. Dabei stellt die Blickrichtung die Rotation des Kopfes um die Gier-Achse dar. Durch gezieltes Ausrichten des Kopfes, werden virtuelle Objekte innerhalb des Displays an der richtigen Stelle positioniert. Befindet sich der Mittelpunkt des Displays innerhalb eines festgelegten Bereiches des virtuellen Objektes, wird das Objekt selektiert.
Um schließlich innerhalb der Applikation navigieren zu können, stellt die Google Glass spezielle Cards bereit, mit welchen Menüs zur eigentlich AR-Applikation entwickelt werden können. Die Bedienung erfolgt hier über Wischgesten am Touchpad des Bügels oder über Sprache. Die Bedienung über Sprache eignet sich besonders für Microinteraktionen, da hierdurch mit wenigen Befehlen die notwendige Aktion abgeleitet werden kann. Nachteilhaft ist, dass die Spracherkennung zum Stand der Arbeit nur in englischer

Sprache möglich ist. Übertragen auf den Anwendungsfall an der Technischen Hochschule, wäre folgender Interaktionsablauf denkbar:
Navigations-Modus

1. Antippen (Aktivschaltung Spracherkennung)
2. „OK, Glass“
3. „Guide me to room <Raum>“ oder „Guide me to lecture <Vorlesung>“

POI-Modus

1. Antippen (Aktivschaltung Spracherkennung)
2. „OK, Glass“
3. „Show room information“

Die Informationen, die im Sichtfeld visualisiert werden, sollten einfach sein und sich zwingend auf das wesentliche beschränken, um den Benutzer nicht mit Informationen zu überfluten. Informationen sollten nur angezeigt werden, wenn sie benötigt werden. Das Konzept sieht vor, nur POIs zu visualisieren, die sich in einem bestimmten Radius befinden, sodass sich die POIs nicht gegenseitig verdecken. Ferner werden nur die wichtigsten Daten in den POIs visualisiert. Durch die Selektion anhand des Blickwinkels, werden nicht interessante Informationen für den Benutzer grundsätzlich ausgeblendet. Nur wenn der Benutzer diese Informationen explizit anfragt, wird ein zusätzliches Kontextmenü eingeblendet. Das Kontextmenü kann beliebige Informationen, wie bspw. aktuelle oder nächste Vorlesung beinhalten.

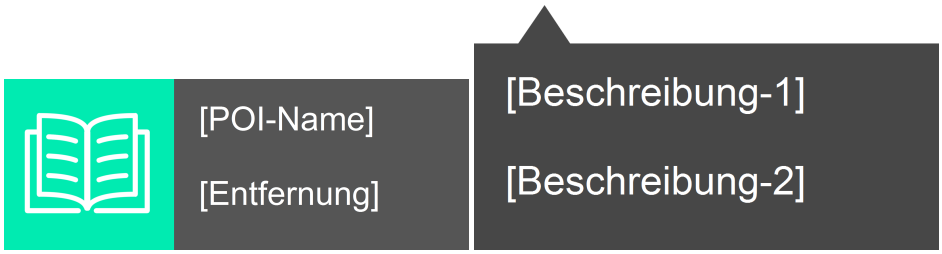

Abbildung 28: Virtuelle Objekte

Die virtuellen Objekte haben dabei folgende Attribute:

1. POI-Name z.B. Raumnummer (2110)
2. Entfernung z.B. 12 m
3. Bild

Das Bild des POIs, soll den POI-Typ durch Symbole eindeutig identifizieren. Auf diese Weise kann der Benutzer verschiedene POI-Typen schnell voneinander unterscheiden.
Informationen sollten immer sowohl im zeitlichen, wie auch im räumlichen Sinne relevant sein. Sie sollten also immer zur richtigen Zeit Kontextabhängig visualisiert werden. Diese Anforderung wurde bereits durch das Grundkonzept realisiert. Hierbei definiert die Position, wie auch der Blickwinkel des Betrachters den Kontext. Zudem werden nur Objekte visualisiert, die in der realen Welt sichtbar sind und nicht durch Hindernisse, wie bspw. Wände verdeckt werden.

4 Implementierung

Mit einem Grundkonzept wurde im letzten Kapitel die Basis für die Implementierung einer Optical-See-Through-basierten Indoor-AR Applikation geschaffen. In folgendem Kapitel wird das bestehende Konzept aufgegriffen und auf dessen Basis die Umsetzung beschrieben. Dabei wird die grundlegende Architektur und die daraus bestehenden Komponenten erläutert.

4.1 Architektur

Grundsätzlich wurde eine klassische 3-Tier Client-Server-Architektur realisiert, wobei ein hybrider Ansatz gewählt wurde. Die Clients sind für die Präsentation der Informationen verantwortlich und dienen als Benutzerschnittstelle. Zugleich beinhalten diese einen Teil der Applikationslogik, um die Serverkomponenten zu entlasten und eine hohe Verfügbarkeit zu unterstützen. Die Server-Komponenten bestehen nur aus Applikationslogik und sind für die Aufbereitung und Weitergabe der Daten verantwortlich. Die Datenbanken persistieren die Daten und sind daher in der Datenschicht einzuordnen.

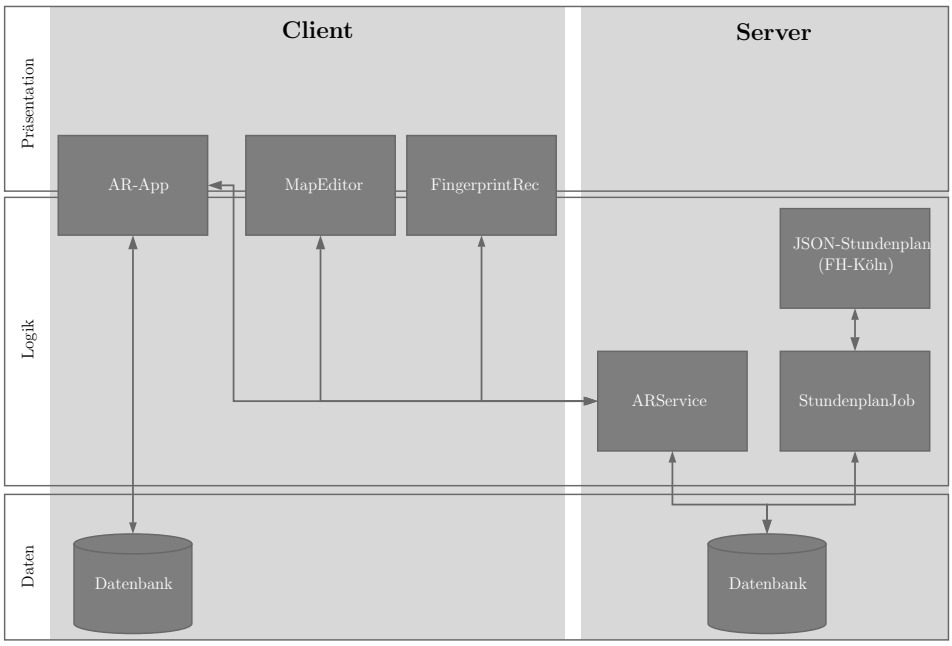

Abbildung 29: Gesamtarchitektur des AR-Systems

Das Ziel der hybriden Architektur ist die permanente Verfügbarkeit des Systems. So ist das System zwar auf eine permanente Online-Nutzung ausgerichtet, jedoch nicht zwingend auf diese angewiesen. Im Normalfall verwendet also die AR-Applikation im Online-Fall alle Daten vom Server, die aus der Applikationslogik heraus geliefert werden. Die gleiche Applikationslogik befindet sich in der AR-Applikation. Bricht die Verbindung ab,

so kann die AR-Applikation mit der eigenen Applikationslogik weiterarbeiten. Dies erfordert allerdings zum einen eine Client-Datenbank, die Server-seitige Daten für eine gewisse Dauer persistieren kann und zum andern einen Synchronisationsmechanismus, der in regelmäßigen Abständen die Daten aus der Server-seitigen Datenbank in die Client-seitige Datenbank übertragt. Dies geht mit dem Nachteil einher, dass viele Daten übertragen werden müssen, also hohe Anforderungen an die Bandbreite gestellt werden. Bei der Wahl der Architektur wurde jedoch die Annahme getroffen, dass sich die Daten nicht regelmäßig ändern. Übertragende Datenobjekte sind hierbei Karteninformationen und Fingerprints, die sich grundsätzlich selten ändern. Lediglich POI-Informationen zu Raumplänen und Vorlesungen werden regelmäßig gepflegt. Hierfür wurde daher ein regelmäßiger Updatezyklus vorgesehen, sodass der Benutzer immer aktuelle Informationen zu POIs erhält.
Die Architektur des Systems setzt sich aus mehreren Komponenten zusammen, die über den AR-Service miteinander verknüpft sind. Im ersten Schritt sammelt die App „FingerprintRec“ die Wifi- bzw. Bluetooth-Fingerprints und übergibt diese an den ARService. Zudem lädt der StundenplanJob in zyklischen Abständen Stundenplandaten aus einem externen Webservice und speichert diese in der Datenbank. Diese Informationen werden an den MapEditor weitergegeben und erlauben die Hinterlegung von POIs auf der Karte. Nachdem Karteninformationen im MapEditor gepflegt wurden, werden diese an den AR-Service übertragen. Der AR-Service nimmt die Karteninformationen entgegen und speichert diese wieder in der Datenbank. Die Kommunikation erfolgt über offene Standards.

4.2 Software-Komponenten

Im letzten Kapitel wurde die Gesamtarchitektur des AR-Systems beschrieben. Im Folgenden soll jede Software-Komponente mit ihren Funktionalitäten und Schnittstellen innerhalb der beschriebenen Architektur im Detail erläutert werden.

4.2.1 ARService

Der ARService stellt die Basiskomponente des gesamten AR-Systems. Sie ist die Schnittstelle zur AR-Datenbank und damit zugleich zwischen allen Applikationen. Zudem enthält der AR-Service einen Teil der Applikationslogik zur Positionsbestimmung mittels Wifi bzw. Bluetooth. Der Service wurde vollständig in Java entwickelt. Zur Nutzung der Webservice-Funktionalität wurde das Jersey-Framework verwendet. Dieses erlaubt das Erstellen von REST-Schnittstellen und verfügt über Annotationen, mit welchen DTO-Objekte einfach serialisiert und deserialisiert werden können.

Zur Kommunikation mit der Datenbank, wurde ein JDBC MySQL Connector eingesetzt.

Der ARService ist ein REST-basierter Webservice. Er weist daher die folgenden Merkmale auf:

1. Jede URI weißt auf eine explizite Ressource
2. Die Daten können in unterschiedlicher Weise repräsentiert werden (in dieser Arbeit im JSON-Format)
3. Die Kommunikation erfolgt zustandslos. Es werden keine Informationen zum Zustand zwischen dem Webservice und dem Client ausgetauscht. Damit ist jede Anfrage an den Webservice stets unabhängig und in sich geschlossen.
4. Der Webservice nutzt bestehende Web-Standards, um die Kommunikation zu ermöglichen. Für den AR-Service wird das HTTP-Protokoll zur Kommunikation mit den Clients verwendet. Dabei werden folgende Operationen unterstützt:

   GET - Bei dieser Operation wird die angegebene Ressource vom Webservice angefragt. Es handelt sich hier lediglich um eine Leseoperation, wodurch die Ressource nicht verändert wird. Beispielsweise werden von der AR-App Karteninformationen angefragt.

   POST - Die Operation POST verändert eine bestehende Ressource. Hierfür werden in der Anfrage die zu ändernden Daten übergeben. Die Informationen werden im JSON-Format übermittelt. Beispiel: Eine Karte wird um zusätzliche POIs erweitert.

   PUT - Legt ein neues Objekt an. Eine Überprüfung von Duplikaten erfolgt nicht und ist nicht notwendig, da beim Anlegen die Karte eine neue eindeutige Identifikationsnummer (ID) erhält. Zum Anlegen eines Objektes müssen die erforderlichen Informationen per JSON in der Anfrage mitgeliefert werden.

   DELETE - Löscht die angegebene Ressource.

Die HTTP-Nachricht unterteilt sich in den Nachrichtenkopf, in welchem die Operation und Informationen zu Kodierungen oder Inhaltstyp übergeben werden, und dem Nachrichtenrumpf, in welchem sich die zu übertragenden Daten befinden. Diese liegen einem REST-Service i.d.R. im JSON-Format vor.
Der ARService verwaltet die Haupt-Entitäten Field, Map, Room, Measurement und Waypoint. Für jede Entität wird sowohl ein eigenes Transferobjekt (DTO) als auch ein Datenzugriffsobjekt (DAO) bereitgestellt. Das Transferobjekt ist ein reines Datenobjekt, welches keine Logik enthält und

nur für die Speicherung der Daten und zur Datenübertragung zwischen dem ARService und den Clients vorgesehen ist. Mithilfe des DTO können die Daten auf einfache Weise serialisiert und deserialisiert werden. Die Voraussetzung ist, dass die DTO's sowohl Client-seitig, wie auch Server-seitig bekannt sind. Die DAO-Objekte stellen die Schnittstelle zur Datenbank dar. Sie stellen Methoden bereit, um die zu den Entitäten entsprechenden Tabellen aus der Datenbank zu lesen und zu schreiben. Diese nutzen ein gemeinsames DatabaseHandler-Objekt, welches die Grundoperationen zur Datenbank bereitstellt und damit eine abstrakte Schicht über das Datenbankschema darstellt.

Die Kommunikation erfolgt über Schnittstellen-Klassen, sogenannte „Ser-

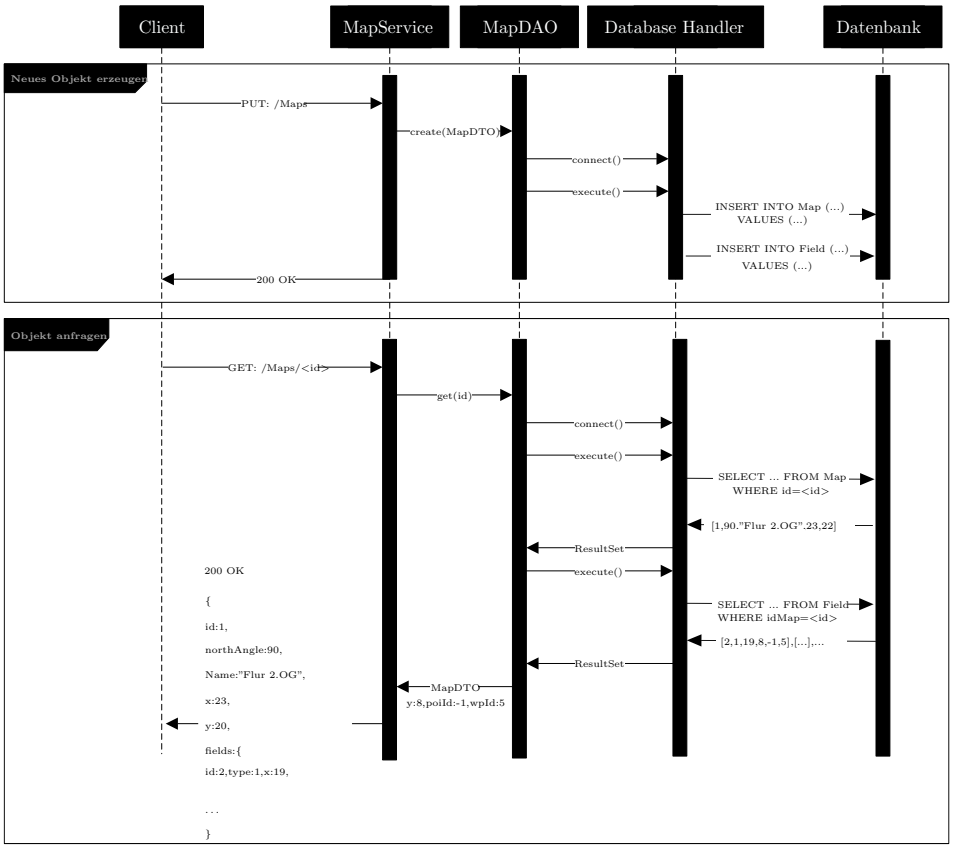

Abbildung 30: Sequenzdiagramm zur Veranschaulichung der Kommunikation mit dem ARService

vices“, die jeweils einer eindeutigen URI zugewiesen werden. Jeder Service wird über die Standart-Operationen GET, POST, PUT, DELETE aufgerufen.

Abbildung 30 veranschaulicht die Kommunikation zwischen dem Client und dem ARService anhand der Entität Map (Karte). Will der Client eine Karteninformation anfragen, führt dieser eine GET-Operation auf den MapService aus. Hierbei wird als Pfad die eindeutige ID übergeben. Der Map-

Service nimmt die ID entgegen und reicht diese weiter an die MapDAO. Die MapDAO nutzt den DatabaseHandler, um zunächst eine Datenbankverbindung aufzubauen. Im zweiten Schritt führt die MapDAO, mithilfe des DatabaseHandlers eine SELECT-Anfrage in der Datenbank aus. Als Filterkriterium wird die ID übergeben. Existiert das Objekt, wird es an die MapDAO als ResultSet zurückgereicht. Die MapDAO erzeugt aus den Daten im ResultSet ein neues MapDTO-Objekt. Der MapService serialisiert das MapDTO-Objekt in das JSON-Format und liefert es in den Nachrichtenrumpf der HTTP-Antwort zurück.
Neben der Datenbankschnittstelle, besitzt der ARService Applikationslogik zur Positionsbestimmung mittels WLAN und Bluetooth. Der ARService stellt hierfür zwei Schnittstellen bereit: StoreWifiService und DeterminePositionService. Der StoreWifiDataService erhält von den Clients eine Liste von MeasurementDTO's, die Messdaten zu WLAN/Bluetooth-Signalstärken enthalten. Ein MeasurementDTO entspricht dabei ein Fingerprint, bestehend aus Signalstärken mehrerer AP's. Für jeden AP wird in der MeasurementDTO ein BeaconDTO Objekt gespeichert. Alle BeaconDTO's werden zunächst in die dafür vorgesehene Datenbanktabelle „Beacon“ abgespeichert. Zudem werden die Feld- und AP-Informationen in separate dafür vorgesehende Tabellen abgelegt.
Nachdem ein neuer Messwert gespeichert wurde, führt der ARService eine Aktualisierung der RadioMap-Tabelle aus. Diese ist eine Aggregationstabelle, die für jeden AP, an jeder Position, für eine bestimmte Blickrichtung und für jede Signalstärke die absolute und relative Wahrscheinlichkeit aggregiert. Die zusätzliche Speicherung der Blickrichtung dient dazu, Absorptionseffekte durch den menschlichen Körper zu reduzieren und damit Ausreißer zu minimieren. Je nach dem, wie oft die die App FingerprintRec zum Einsatz kommt, wird die RadioMap stetig aktualisiert. Dies geschieht immer, wenn ein neuer Messwert aufgezeichnet wurde. Auf diese Weise stellt die RadioMap-Tabelle stets ein aktuelles Histogramm aller Signalstärken eines AP's an einer bestimmten Position bereit, welches in der Online-Phase abgefragt werden kann. Die gesamte Berechnungs- bzw. Aggregationslogik wurde daher in die Datenbank verlagert, da davon ausgegangen wurde, dass die Datenbank Operationen mit sehr vielen Daten bereits hochperformant durchführt. Im Anhang kann der dafür benötigte SQL-Quellcode eingesehen werden.
Zunächst wird die Anzahl der Stichproben N pro Feld und AP berechnet und ein hinreichend kleiner λ-Wert gesetzt. Anschließend werden aus der Beacon-Tabelle alle Häufigkeitsverteilungen der aufgetretenen Signalstärken nach AP, Position und Blickrichtung gruppiert, aggregiert und mit einer statischen Tabelle Space, die alle Signalstärken enthält, ein Join ausgeführt, damit alle Signalstärken im Histogramm vertreten sind. Null-Werte werden

mit *lambda* belegt. Im zweiten Schritt wird das erzeugte Histogramm in der Tabelle RadioMap gespeichert.
Der DeterminePositionService wird mit einem MeasurementDTO aufgerufen, der in der Online-Phase erzeugt wurde. Das übergebene Objekt enthält mehrere BeaconDTO's die Informationen zur MAC-Adresse des AP's bzw. Bluetooth-Beacons, die Blickrichtung und die aufgezeichnete Signalstärke enthalten. Der dazugehörige SQL-Code ist ebenfalls im Anhang zu finden. Im ersten Schritt iteriert der DeterminePositionService durch alle BeaconDTO's und sammelt die Einzelwahrscheinlichkeiten aller möglichen Felder zu der MAC-Adresse, Signalstärke und Orientierung des jeweiligen Beacons in der RadioMap-Tabelle. Im zweiten Schritt werden die Daten aggregiert, in dem die Einzelwahrscheinlichkeiten von gleichen Positionen miteinander multipliziert werden. Da Standard-SQL keine Multiplikationsfunktion anbietet, wurde dies mithilfe der Logarithmus- und Exponentialfunktion gelöst. Auf diese Weise kann schließlich die Gesamtwahrscheinlichkeit aller Positionen errechnet werden. Die Liste aller Gesamtwahrscheinlichkeiten wird schließlich an den Client zurückgegeben.

4.2.2 FingerprintRec

FingerprintRec ist eine im Rahmen dieser Arbeit, mit dem Android SDK entwickelte, Smartphone-Applikation (App), die dazu dient an verschiedenen Positionen Bluetooth- bzw. WLAN-Fingerprints zu sammeln und diese an den AR-Service zu übertragen. Auf diese Weise können Fingerprint-Daten zur Offline-Phase gespeichert werden, die später in der Online-Phase zur Positionsbestimmung herangezogen werden können.

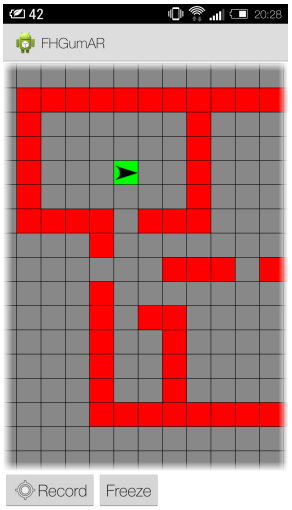

Abbildung 31: Haupt-Activity der FingerprintRec-Applikation

Beim Starten der App stehen dem Benutzer alle verfügbaren Karten zur Auswahl. Wurde eine Karte ausgewählt, wird diese vom ARService heruntergeladen und visualisiert (siehe Abbildung 31). Über Toucheingaben, kann der Benutzer sich in alle Richtungen auf der Karte bewegen. Zum Aufnehmen von Fingerprints, drückt der Benutzer auf die jeweilige Position und auf den Button „Record“. Das Programm beginnt damit in zyklischen Abständen nach Bluetooth bzw. WLAN-Beacons zu horchen und bei Empfang dessen Quell-MAC-Adresse, empfangene Signalstärke und die aktuell ausgewählte Position in einer Liste von BeaconDTO's zu speichern. Des Weiteren wird die Blickrichtung des Benutzers ermittelt und als zusätzliche Information jedes Beacons im BeaconDTO-Objekt gespeichert. Dabei werden nur die BeaconDTO-Objekte in die Liste aufgenommen, die im Blickwinkel (0° bzw. 360°,45°,90°,135°,180°,225°,270°,315°) aufgezeichnet wurden. Diese Einschränkung erlaubt später bei der Positionsbestimmung eine Aggregation über ähnlich viele Messdaten. Dabei wurde eine Toleranz von ±8 konfiguriert, um Schwankungen der Inertialsensorik zu kompensieren.

Der Scan geschieht so lang, bis der Benutzer die Aufnahme abbricht. Unterbricht der Benutzer die Aufnahme, werden die Beacons als MeasurementDTO an den ARService übertragen. Die Applikation verwendet hierfür die StoreWifiService-Schnittstelle des ARService, um die Objekte in der Datenbank zu speichern.

4.2.3 Stundenplan-Job

Zur Visualisierung der Rauminformationen der Technischen Hochschule Köln am Campus Gummersbach, werden Informationen zu Vorlesungen, wie Uhrzeit, Dauer, Vorlesungsraum und ggf. Dozent benötigt. Da diese Informationen nicht ohne Weiteres in der Datenbank zur Verfügung stehen, müssen diese extern beschafft werden. Die TH-Köln hat hierfür einen Webservice bereitgestellt, der zur Abfrage der Stundenpläne am Campus Gummersbach genutzt werden kann. Zum Stand der Arbeit lieferte der Webservice allerdings immer nur alle Vorlesungen als JSON-Objekt zurück (siehe dazu Anhang). Zudem lagen die Daten in einer denormalisierten Form vor, wodurch die spätere Kombination mit Feldern und dessen Abfrage erschwert geworden wäre. Um die Datenmenge für mobile Geräte möglichst klein zu halten und normalisierte Daten zu speichern, wurde ein Stapelverarbeitungsprogramm entwickelt, welches in zyklischen Abständen die Stundenplan-Daten aus dem Stundenplan-Webservice anfragt und in die AR-App-eigene Datenbank überträgt. Der Stundenplan-Job wurde vollständig in Java entwickelt. Er besteht aus einer Schnittstelle WSInterface,

die für die HTTP-Kommunikation mit dem Webservice verantwortlich ist und einer Datenbankschnittstelle DatabaseHandler, die eine Verbindung zur Datenbank aufbaut und die Daten per SQL-Insert in die Datenbank speichert. Das WSInterface stellt eine HTTP Anfrage an den Webservice, erhält eine Liste von JSON-Objekten und zerlegt diese in DTO-Objekte. Die Daten werden an den DatabaseHandler übergeben. Hier werden die Daten nochmals normalisiert und per INSERT-Befehl in die beiden Tabellen Room und Lecture übertragen. Auf diese Weise können sie per Foreign-Key-Beziehung mit den jeweiligen Feldern verknüpft werden. Die Applikation ist auf dem AR-Service eigenen Server lokalisiert und ist daher befugt direkt auf die interne Datenbank zuzugreifen.

4.2.4 MapEditor

Der MapEditor ist eine mit Java-Swing entwickelte, Dialog-basierte Applikation, die dazu dient die Karteninformationen zu pflegen, die von der AR-App vorausgesetzt werden. Der Editor ermöglicht dabei bestehende Raumpläne in das von der AR-App erforderliche Koordinatensystem zu überführen. Im ersten Schritt lädt der Benutzer ein Bild eines Raumplanes und wählt die Rastergröße. Das Programm teilt das Bild schließlich in ein zweidimensionales Raster. Innerhalb des Rasters können schließlich folgende Informationen hinterlegt werden:

1. Freifeld - Jedes Feld, das nicht belegt wurde ist grundsätzlich ein Freifeld. Für die spätere Berechnung der kürzesten Pfade zwischen Waypoints bzw. die Berechnung der Peilung zur POIs haben diese Felder keine Bedeutung. Aus diesem Grund werden diese Felder nicht zwischen dem AR-Service und dem MapEditor bzw. der AR-App ausgetauscht. Hierbei reicht die Information, wieviel Felder ein solches Raster besitzt. Der Benutzer kann ein durch Hindernis, POI bzw. Waypoint belegtes Feld jederzeit durch ein Freifeld ersetzen.

2. Hindernis - Mithilfe von Hindernissen, können Wände und andere Gegenstände innerhalb des Koordinatensystems platziert werden. Diese sind vor allem für die Berechnung des kürzesten Pfades zwischen zwei Wegpunkten relevant. Ferner wird die Information genutzt, um in der AR-App POIs hinter Wänden oder anderen Gegenständen auszublenden, um den Benutzer nicht mit unnötigen Informationen zu stören. Hindernisfelder werden in der Regel also an den Stellen gesetzt, wo der importierte Raumplan Wände und Türen enthält. Auf diese Weise kann schnell und intuitiv ein Umgebungsmodell konstruiert werden. Jedes Hindernisfeld wird mit roter Farbe visualisiert und lässt sich durch Klick auf das jeweilige Feld setzen.

3. POI - Dieser Feldtyp dient dazu Informationen zu POIs innerhalb des Koordinatensystems zu hinterlegen. Jeder POI enthält die Informationen wie Raum-Nr., Raum-Bezeichnung, aktuelle Vorlesung etc. Zu den als POI markierte Felder werden später die jeweilige Peilung berechnet und je nach Blickwinkel des Betrachters als virtuelle Objekte im Sichtfeld visualisiert. Möchte der Benutzer ein neues POI im Raster hinterlegen, wählt er eines aus der Liste im Rechten Bereich (siehe Abbildung 32) aus und klickt auf das jeweilige Feld. Das POI wird in der Liste blau markiert und kann nicht ein zweites mal in die Karte eingefügt werden. Soll dieses wieder entfernt werden, steht dem Benutzer das Freifeld zur Verfügung.

4. Waypoint - Das Waypoint-Feld erlaubt es Wegpunkte für die spätere Navigation und damit die Laufwege innerhalb des Koordinatensystems festzulegen. Die Wegpunkte werden später in einen Graphen überführt und schließlich der kürzeste Weg von einem Wegpunkt zum anderen ermittelt. Um einen Wegpunkt zu erzeugen, selektiert der Benutzer den Feldtyp und klickt auf ein beliebiges Feld. Das Feld wird zunächst gelb markiert. Bewegt der Benutzer den Zeiger über ein anderes Feld wird der von einem Pathfinding-Algorithmus ermittelte Pfad vom Quell- zum Zielpunkt ermittelt. Die gelbe Markierung stellt den kürzesten Laufweg zum nächsten Wegpunkt dar. Diese dient als Hilfestellung, um jeweils den nächsten Wegpunkt auf einem geraden Laufweg zu platzieren. Auf diese Weise können mit wenigen Wegpunkten schnell alle wichtigen Laufwege abgebildet werden.

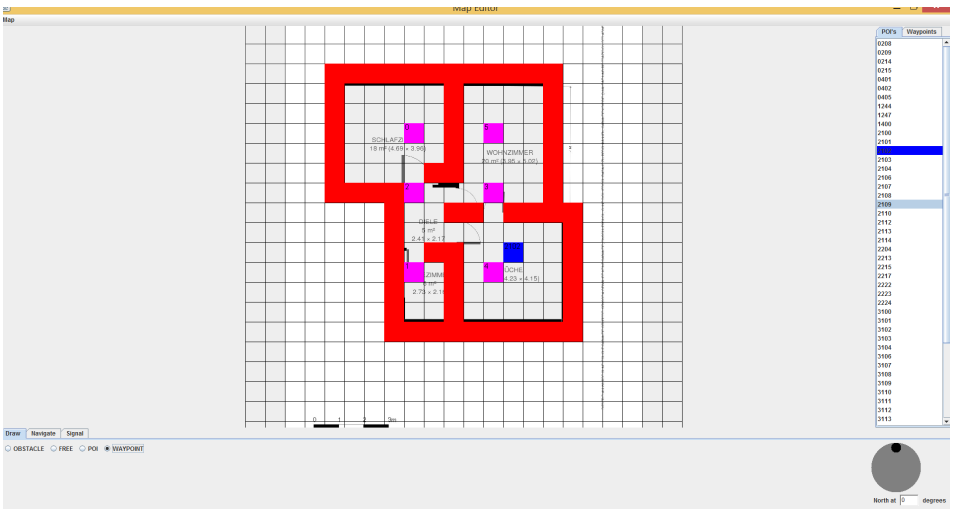

Abbildung 32: Mapeditor zur Pflege der Karteninformationen

Jeder Wegpunkt besitzt eine eindeutige Identifikationsnummer (ID). Diese ID spiegelt den Primärschlüssel in der Datenbank dar. Jede ID wird

also mit einer eindeutigen Position in der Datenbank gespeichert. Um die LLAMarker basierte Positionsbestimmung zu unterstützen, wurde der MapEditor um einen Marker-Generator erweitert. Die ID wird vom Marker Generator in einen 5x5 großen Marker kodiert. Der Marker-Generator besitzt eine Druck-Funktion, um den generierten Marker auf einem DIN-A4 Blatt auszudrucken. Auf diese Weise kann der Marker später an den jeweiligen Stellen platziert und von der AR-App eingescannt und dekodiert werden.
Neben dem Marker-Generator und den Feldtypen, verfügt der MapEditor im unteren rechten Bereich über ein rundes Steuerelement, mit der die Himmelsrichtung der Karte angegeben werden kann. Die Angabe der Himmelsrichtung ist für die spätere Ermittlung der Peilung relevant, da sie das δ, also die Deklination zwischen magnetischen Norden und dem relativen Nordens im erzeugten Gitternetz darstellt. Hier sei nochmals auf die Formel 10 verwiesen.
Des Weiteren besitzt der Editor eine Speichern- und Laden-Funktion, mit der die gesamte Karte mitsamt der Informationen in der Datenbank gespeichert wird. Zum Speichern und Laden der Karteninformationen dient der AR-Service als Schnittstelle zur Datenbank. Gespeichert werden die Informationen

1. Name der Karte
2. Himmelsrichtung
3. Alle nicht leeren Felder d.h. Waypoints, POIs und Hindernisse
4. Rastergröße - Anhand der Rastergröße müssen nicht alle Felder in der Datenbank gespeichert werden, wodurch schließlich auch weniger Daten übertragen werden müssen. Durch Kombination der Rastergröße und den nicht leeren Feldern kann so die vollständige Karte am Client rekonstruiert werden.

Zuletzt können mithilfe der Navigationsfunktion alle Laufwege getestet werden. Bei Klick auf den Reiter „Navigate“ im Menü, schaltet die Applikation in den Navigationsmodus. Im ersten Schritt markiert der Benutzer durch Klick auf einen beliebigen Waypoint die Startposition. Der Waypoint erscheint grün. Im zweiten Schritt klickt der Benutzer auf die Zielposition. Mithilfe des Dijkstra Algorithmus wird nun der kürzeste Weg über die Waypoints ermittelt. Jeder zu besuchende Waypoint wird grün markiert. Der Algorithmus greift nur, wenn die Waypoints untereinander auf einer Linie angeordnet sind. Auf diese Weise kann man schließlich die gesetzten Laufwege überprüfen.

4.2.5 AR-App

Die AR-App stellt die Hauptkomponente des AR-Systems dar, da sie die wichtigste Applikationslogik zur Realisierung der erweiterten Realität bereitstellt. Die Anwendung wurde auf Android 4.4, der Auslieferungsplattform der Google Glass entwickelt. Ferner wurde zum entwickeln das von Google bereitgestellte Glass Development Kit (GDK) verwendet. Das GDK erweitert das aus Android bereits bekannte Android SDK um neue UI-Elemente wie Cards oder Immersions, wie auch erweiterte Funktionen, wie etwa die Spracherkennung oder Touchgesten. Das Android SDK enthält bereits eine Reihe von Programmierschnittstellen und Bibliotheken, um die Hardware der Google Glass ansprechen zu können.

Anders als die klassische Android App, die nach der Installation automatisch registriert wird und damit die Haupt-Activity [90] aus dem Hauptmenü heraus aufgerufen werden kann, steht bei einer GDK-App stets die Timeline im Vordergrund. Der Benutzer ruft die Home-Karte durch Antippen des Displays auf, von wo aus per Sprache bzw. Touch-Gesten die App aufgerufen wird. Hierfür muss ein sogenannter Intent-Filter registriert werden. Damit wird dem Betriebssystem mitgeteilt, dass die App anhand eines Intents[91] gestartet werden kann. Der Intent-Filter verwendet eine sogenannte Voice-Trigger-Action, zum Aufruf der App per Sprache über die Home-Karte. Das GDK legt zudem als Rückgriff einen Menüpunkt im Home-Kontextmenü mit dem Namen der App an, somit kann der Benutzer die App auch per Touchgesten ausführen.
Die Home-Karte verwendet einen Intent um den AppService zu starten. Dieser läuft im Hintergrund und dient dazu, die App jederzeit zu pausieren und wieder aufzunehmen, wenn der Benutzer diese benötigt. Der App Service hat die Aufgabe, die Hauptkarte der App zu verwalten d.h. zu erzeugen und zu entfernen, wenn nötig. Für die AR-App wurde eine Immersion als Hauptkarte gewählt, da sie unabhängig von der Timeline alle UI-Komponenten der Android API bereitstellt und daher den höchsten Grad an Anpassungsmöglichkeiten zur interaktiven Erfahrung bietet. Die Immersion ist im wesentlichen eine Activity, die bereits durch das Android SDK bereitgestellt wird. Der Service startet also die MainActivity welche wiederum mit der AR-Activity für das Rendering der virtuellen Objekte zuständig ist und zudem eine MenuActivity, die das Menü beinhaltet.

Die Applikation umfasst folgende Aufgaben bzw. Funktionalitäten:

[90] Stellt eine Bildschirmseite, und daher die wichtigste Komponente in Android dar.

[91] Intents werden aus dem Android SDK bereitgestellt und dienen dazu Funktionalitäten über Applikationskomponenten auszutauschen. Oft werden sie dazu genutzt, Activities untereinander zu starten. Dafür muss die Applikation einen Intent-Filter registriert haben.

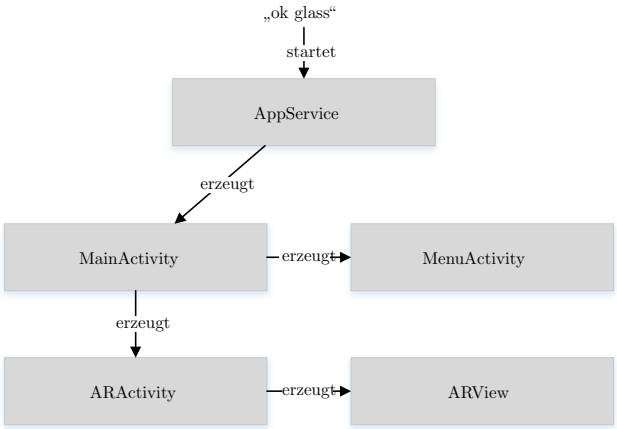

Abbildung 33: Google Glass Architektur

1. Positionsbestimmung
2. Darstellung von POIs mit Optical-See-Through-Technik
3. Navigation
4. Schnittstelle zur Benutzer-Interaktion

Zur Positionsbestimmung stellt die Android-API bereits sogenannte LocationManager bereit, die mittels WLAN, Mobilfunk und GPS die Position ermitteln können. Diese sind allerdings hauptsächlich für den Outdoor-Bereich konzipiert und können daher für die AR-Applikation nicht genutzt werden. Das System wurde daher um drei zusätzliche LocationManager erweitert. Diese setzen voraus, dass mithilfe des MapEditors bereits Karteninformationen gepflegt wurden und der AR-Service als Schnittstelle fungieren kann:

WifiLocationManager
Der WifiLocationManager zeichnet alle drei Sekunden die empfangenen WLAN-Beacons der umgebenden AccessPoints auf und ermittelt anhand der Sensoren die Blickrichtung des Benutzers. Für das Scannen der umgebenden Access Points stellt das Android SDK den WifiManager bereit. Der OrientationManager hat die Aufgabe alle Sensoren miteinander zu fusionieren und daraus einen Rotationsvektor zu bestimmen. Dieser verwendet den SensorManager des Android SDK's. Schließlich wird aus den Informationen AP, Signalstärke und Orientierung je ein Beacon-Objekt erzeugt und in eine Liste eingefügt. Die Liste wird an den AR-Service übergeben, welcher eine Liste von möglichen Positionen mit ihrer Eintrittswahrscheinlichkeit zurückliefert. Ferner enthält der LocationManager eine Liste von WifiListener, für welche sich die Applikation, d.h. die Hauptklasse ARActivity registrieren kann. Wird eine neue Position zurückgeliefert, werden die

Listener benachrichtigt. Auf diese Weise kann bei einem Positionswechsel immer eine Aktion erfolgen. Um diesen Mechanismus zu implementieren, wurde das Observer-Pattern[92] angewandt.

BluetoothLocationManager
Der BluetoothLocationManager sammelt alle drei Sekunden empfangene Bluetooth Beacons und die Blickrichtung und sendet diese, analog zum WifiLocationManager, als Liste von Beacon-Objekten zum AR-Service. Auch dieser Manager verwendet Listener, um die ARActivity bei Positionswechsel zu benachrichtigen.

LocationTagManager
Der LocationTagManager verwendet LLA-Marker um die Position des Benutzers zu bestimmen. Während bei den Funk-basierten Methoden die Positionserkennung in regelmäßigen Abständen automatisiert durchgeführt wird, muss diese über LLA-Marker manuell angestoßen werden. Dies kann durch das Hauptmenü der Applikation geschehen. Wählt der Benutzer diese Option, wird über einen sogenannten Intent eine neue Activity erzeugt, die den Benutzer auffordert einen Marker einzuscannen. Die Kamera der Google Glass nimmt zu diesem Zeitpunkt permanent das Bild auf. Die Erkennung des Markers und das Ermitteln der kodierten eindeutigen Id erfolgt über das ArUco-Framework. Ist die Id bekannt, wird diese an den AR-Service übergeben. Der AR-Service ermittelt schließlich aus der Id eine eindeutige Position und liefert diese an die AR-Applikation zurück. Die gesamte Kommunikation mit dem AR-Service erfolgt über den WSHandler. Dieser baut eine Verbindung mit dem Webservice auf und führt Operationen auf diesen aus.

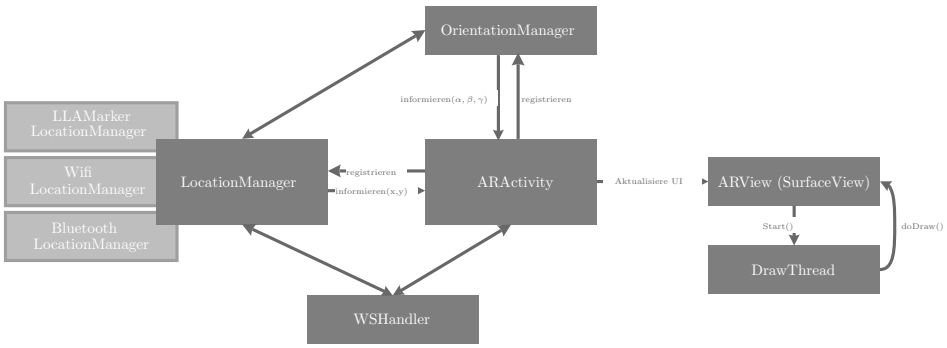

Abbildung 34: Hauptkomponenten der AR-App und deren Zusammenspiel

Nachdem die Position ermittelt wurde, wird diese nochmals von der

[92] Vgl. Gamma et al., *Design Patterns - Elements of Reusable Object-Oriented Software*, S.326

ARActivity über den WSHandler an den AR-Service übergeben, um die dazugehörige Karte zu ermitteln. Die vom AR-Service zurückgeliefert Karte enthält alle vom MapEditor erzeugten Felder mit hinterlegten Informationen wie Hindernisse, Waypoints und POIs als DTO-Objekte. Die ARActivity wandelt schließlich die mit POI hinterlegten Datenobjekte in View-Objekte mit den folgenden Informationen um:

1. Bild
2. Rotation
3. Bildschirmkoordinaten (x,y)
4. Sichtbarkeit (ja/nein)
5. Breite/Höhe

Die View-Objekte enthalten alle notwendigen Informationen, die zum Zeichnen der POI-Objekte auf dem Bildschirm vorgesehen sind. Die Rotation wird direkt aus dem Roll-Winkel ermittelt, die Bildschirmkoordinaten nach Formel 11 und 12. Das Attribut Sichtbarkeit legt fest, ob sich das Objekt hinter oder vor einem Hindernis befindet. Die Breite und Höhe eines View-Objekts ist zunächst statisch und wird je nach Distanz zum entsprechenden POI skaliert. Bei der Skalierung wird das Seitenverhältnis beibehalten. Das POI-Bild ist statisch und wird je nach POI-Typ gewählt.
Nachdem die View-Objekte erzeugt worden sind, werden sie an die AR-View übergeben. Diese ist für den gesamten Zeichenprozess verantwortlich. Sie erweitert die Android SurfaceView, die dazu dient, UI-Elemente außerhalb des Haupt-UI-Thread bearbeiten und anzeigen lassen zu können. Zudem ist die SurfaceView für rechenintensive Animationen optimiert und unterstützt bei der Zeichnung der POIs hohe Bildwiederholungsraten zu erreichen und das Bild flüssig wirken zu lassen. Die AR-View verwendet einen eigenen Thread „DrawThread“, der in einer Schleife regelmäßig die doDraw()-Methode der ARView aufruft, mit welcher anhand der neuen Informationen der POIs, wie Rotation, Bildschirmkoordinaten etc. die Objekte auf der SurfaceView neu gezeichnet werden.

Die Navigation erfolgt über den NavigationProvider. Dieser wird zunächst mit allen Wegpunkten, die im MapEditor definiert wurden initialisiert. Im ersten Schritt erzeugt der NavigationProvider aus den Wegpunkten einen nicht-negativen kantengewichteter Graph, bestehend aus Kanten und Knoten, wobei die Knoten die Wegpunkte und die Kanten die Wege zum den jeweiligen Wegpunkten speichern. Die Kantengewichtungen werden anhand der Entfernung zwischen den jeweiligen Wegpunkten ermittelt. Diese errechnet sich durch die euklidische Distanz. Es werden nur die Wegpunkte

mit einer Kante verknüpft, die durch eine Gerade ohne Winkeländerung abbildbar sind und keine Hindernisse auf der Strecke enthalten.
Zur Berechnung des kürzesten Pfades wird dem Provider ein Ursprungswaypoint und eine Zielposition übergeben. Im ersten Schritt ermittelt der NavigationProvider den nächsten Wegpunkt zur übergebenen Zielposition. Damit hierbei auch die tatsächliche Distanz erechnet werden kann, müssen Hindernisse berücksichtigt werden. Daher verfügt der NavigationProvider über einen Dijkstra-Algorithmus, der alle Wegpunkte durchläuft, die Pfadlänge vom jeweiligen Wegpunkt zur Zielposition errechnet. Der Wegpunkt mit der kürzesten Pfadlänge wird selektiert.
Im zweiten Schritt berechnet der NavigationProvider den kürzesten Pfad zwischen dem Ursprungs-Wegpunkt und dem Ziel-Wegpunkt. Die Berechnung erfolgt über einen Dijkstra-Algorithmus, welcher auf den vorher erzeugten Graph ausgeführt wird. Der Pfad wird als Liste von Feldern zurückgegeben. Relevant für die Visualisierung eines Wegpunktes ist schließlich das erste Element der Liste. Wird ein neuer Wegpunkt besucht, startet der NavigationProvider den gesamten Prozess neu und berechnet einen neuen Pfad.

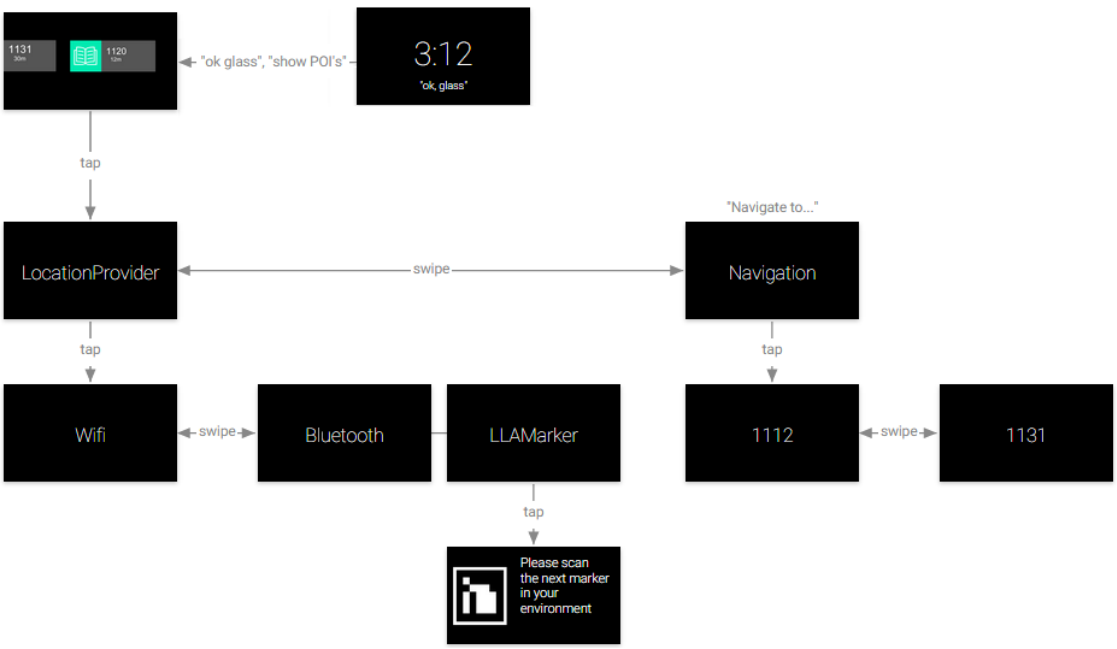

Abbildung 35: Ablauf der AR-App im Glassware Flow-Designer

Die Bedienung der AR-App gestaltet sich durch die Verwendung der Google Glass Cards. Zum Starten der App tippt der Benutzer auf die Touchoberfläche der Google Glass. Es erscheint der gewohnte Startbildschirm, von welchem die Applikation entweder per Sprache oder Auswahl gestartet werden kann. Beim Starten der App wird zunächst die Haupt-

activity geöffnet und der POI-Modus aktiviert, bei welchem alle POIs der Umgebung visualisiert werden. Ferner wird standardmäßig der LLAMarker-LocationProvider zur Bestimmung der Position verwendet. Durch Tippen auf die Touchoberfläche, erhält der Benutzer ein Kontextmenü, in welchem er den LocationProvider auswählen oder den Navigationsmodus aktivieren kann. Die Navigation zu einem Raum kann zudem per Sprache („Navigate to ...“) erfolgen.

4.3 Datenbank

Zur persistenten Speicherung der Hauptentitäten wird eine MySQL-Datenbank eingesetzt. Diese kommuniziert direkt mit dem ARService. Die Struktur der Datenbank wird im ER-Diagramm in Abbildung 36 veranschaulicht.

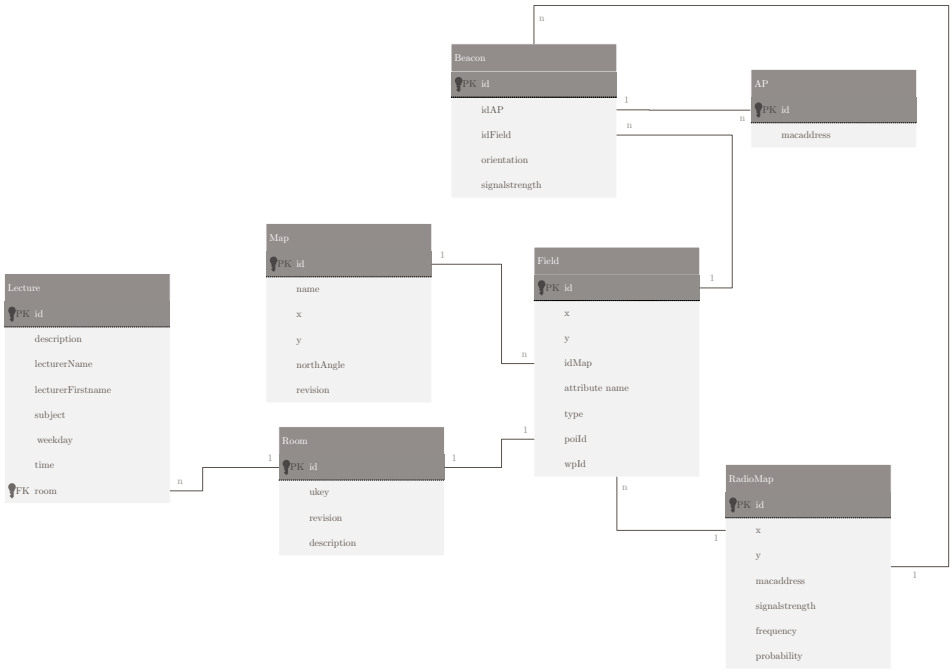

Abbildung 36: ER-Diagramm der Datenbankstruktur

Zur Speicherung der Kartendaten sind vor allem die Tabellen Map und Field relevant. Die Tabelle Map enthält u. a. Informationen zur Rastergröße und der Himmelsrichtung der Karte. Letzteres ist für die Bestimmung der Peilung unabdingbar. Die Tabelle Map besitzt eine Menge von Feldern, die jeweils durch ihre Position innerhalb des Rasters und der dazugehörigen MapId eindeutig identifiziert wird. Diese speichert zudem den Feldtyp und je nach Feldtyp eine eindeutige POI-ID und eine Waypoint-ID. Der POI ist in diesem Fall ein Raum dem wiederum mehrere Vorlesungen zugewiesen sind.
Zur Positionsbestimmung wurden zusätzlich die Tabellen Beacon, AP und RadioMap erzeugt. Jeder Beacon stammt von einem bestimmten AP, wurde

an einem bestimmten Feld und in einer bestimmten Orientirung aufgezeichnet und besitzt eine Signalstärke. Die Orientierung wird gespeichert, um so die spätere Filterung nach ähnlichen Orientierungswinkeln zu ermöglichen. Für jeden AP existiert eine eigene Tabelle, die jeweils nur die MAC-Adresse einmalig speichert. Auf diese Weise erhält man eine normalisierte nichtredundante Form. Eine Ausnahme bildet die Tabelle „RadioMap“. Um eine hohe Performance bei der Server-seitigen Positionsbestimmung zu ermöglichen, werden die Daten aus Field, Beacon und AP in einer gemeinsamen Tabellen zusammengeführt. Dabei werden die Signalstärken aller Beacon-Datensätze mit der gleichen Feld-ID, AP-ID und Signalstärke aggregiert, um die relative Wahrscheinlichkeit zu berechnen. Auf diese Weise kann der ARService die Wahrscheinlichkeiten einer Signalstärke pro Feld und AP direkt auslesen und bedarf zur Online-Phase keiner weiteren Applikationslogik. Neben den genannten Tabellen wurde zusätzlich eine View „Space“ erzeugt, die alle Signalstärken von -100 bis 0 auflistet. Diese ist allerdings nur eine Hilfstabelle, die dazu dient, bei der Aggregation der Datensätze in der RadioMap-Tabelle alle Signalstärken zu berücksichtigen, auch wenn keine Daten zu Signalstärken existieren. Auf diese Weise ist immer garantiert, dass für jeden AP, jedes Feld und jede Orientierung ein eigenes Histogramm mit eigener Häufigkeitsverteilung gespeichert wird. Dies erfordert hohen Speicherplatz und Rechenaufwand in der Offline-Phase, verringert jedoch den Rechenaufwand in der Online-Phase.

5 Evaluation

Nachdem das Konzept und die Implementierung der AR-Anwendung beschrieben wurde, soll in diesem Kapitel die finale Lösung nochmals evaluiert werden. Dabei sollen vor allem die Grundvoraussetzungen eines AR-System und damit die wichtigsten Grundkomponenten und Funktionen untersucht werden. Die Evaluation fand in Form eines Praxistests statt, bei welchem eine vollständige Testumgebung konstruiert wurde. Hierfür wurde eine $70m^2$-große Fläche in ein 20x26-großes Regular-Grid unterteilt. Zur Erstellung des Regular Grids wurde im MapEditor der Raumplan importiert und darüber ein Gitter mit der Gittergröße 1 m erzeugt. In Abbildung 37 ist zu

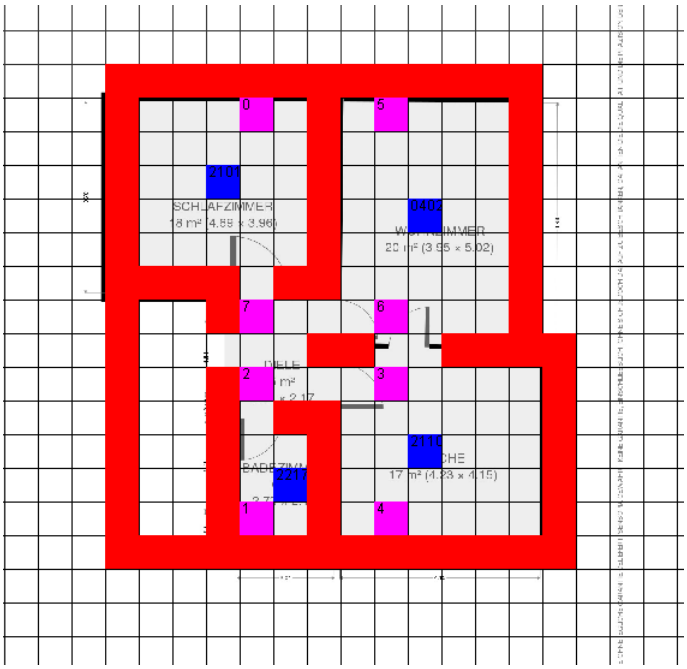

Abbildung 37: Versuchsaufbau im MapEditor

erkennen, dass im MapEditor die Wände, als Hindernisse markiert wurden. Zudem bilden acht Wegpunkte die Laufwege auf der gesamten Fläche. Die Wegpunkte sind so platziert, dass gerade Laufwege ermöglicht werden. Insgesamt unterteilt sich die Fläche in vier Räumen, die durch Wände voneinander getrennt sind. Jeder Raum wurde im Koordinatensystem als eine POI abgebildet.

Für den Praxistest wurden zwei Testreihen durchgeführt. Für die erste Testreihe wurden zur Positionsbestimmung WLAN-Fingerprinting eingesetzt, um damit die Tauglichkeit der Fingerprint-Methode für eine ortsbasierte AR Anwendung zu untersuchen. Bei der zweiten Testreihe wurden LLA-Marker eingesetzt, um die AR-Anwendung im Einsatz mit einer exakten Positionsbestimmung zu untersuchen. Für beide Testreihen wurden

folgende Kriterien evaluiert:

1. Funktioniert die Positionsbestimmung? Ist damit eine präzise Anordnung von virtuellen Objekten möglich?

2. Wenn die Position korrekt bestimmt wurde und virtuelle Objekte korrekt positioniert wurden: Sind die angezeigten Informationen Kontextabhängig, d.h. haben sie einen Bezug zur aktuell wahrgenommenen Realität?

3. Wie hoch ist der Grad der Verschmelzung? D.h.: Werden die virtuellen Objekte korrekt Registriert und zur Realität überlagert ? Hierbei sollten die Grundaspekte eines AR-Systems berücksichtigt werden:

   - Registrierung

   - Tracking

   - Darstellung & Ausgabe

4. Erfüllt die Navigation ihren Zweck - D.h. wird immer der nächste Wegpunkt zum Ziel visualisiert? Wird das gewünschte Ziel erreicht?

Im ersten Schritt wurden die oben genannten Kriterien in Verbindung mit dem WLAN-Fingerprint-Verfahren untersucht. Hierbei wurden in der Offline-Phase an jeder 2. Position über die Laufwege, 20 Scandurchläufe pro Richtung (0°,90°,180°,270° Grad), also insgesamt 16 Positionen aufgezeichnet. Schließlich wurden die Testreihe mithilfe der AR-App durchgeführt. Dabei wurden im POI-Modus an verschiedenen Positionen (siehe Abbildung 37) die POIs in der Umgebung betrachtet. Hierbei konnte man feststellen, dass die Positionsbestimmung, nach Erhöhung der Feldanzahl und Reduzierung der Scandurchläufe tendenziell etwas schlechter ausfiel, jedoch immer noch im Rahmen der 1-2 Meter-Grenze. Es fiel zudem auf, dass die Positionsbestimmung an Instabilität litt, die sich durch häufige Sprünge der virtuellen Elemente auf dem Bildschirm bemerkbar machte. Nur wenige Sprünge führten allerdings dazu, dass die virtuellen Elemente vollständig verschwanden. Diese Sprünge wurden als sehr störend empfunden und sind für ein ortsbasiertes AR-System nicht tragbar. Man kann daher zusammenfassen, dass zwar die Fingerprinting-Methode für den Einsatz eines ortsbasierten AR-Systems eine denkbare Lösung ist, jedoch noch weiterer Optimierungen bedarf, um eine Präzise Positionierung von virtuellen Objekten zu ermöglichen.
Schließlich wurde die AR-App mit Markern getestet. Für jeden Waypoint wurde hierfür zunächst im MapEditor ein Marker generiert. Anschließend wurden diese ausgedruckt und an den jeweiligen Positionen, entsprechend ihrer ID, auf der Fläche verteilt. Die Marker wurden auf dem Boden positioniert, so dass eine metergenaue Positionsbestimmung möglich ist. Die

Positionsbestimmung mit Markern wurde insgesamt als sehr stabil bewertet, da in in 18 von 20 Fällen, der Marker korrekt erkannt, die ID an den ARService übergeben und schließlich die Position vom ARService richtig zurückgegeben wurde. In den zwei anderen Fällen wurde die Markererkennung vor allem bei schlechten Lichtverhältnissen getestet. Gerade bei sehr hellen bzw. sehr dunklen Umgebungen werden sehr kontrastarme Bilder erzeugt, die durch eine Normalisierung nicht mehr aufzuheben sind. In diesen Fällen ist die AR-App nicht in der Lage die Position korrekt zu bestimmen.

In beiden Fällen, der Fingerprint-Methode und Markererkennung, bei denen die Position korrekt bestimmt wurde, wurden die virtuellen Objekte in den meisten Fällen kontext-abhängig visualisiert, d.h. alle Räume wurden korrekt erkannt und mit entsprechenden POI-Informationen versehen. Es wurden hier allerdings Einzelfälle beobachtet, bei denen die virtuellen Elemente leicht versetzt positioniert wurden. Es stellte sich heraus, dass der Magnetsensor in der Nähe von metallischen Gegenständen oder elektrischen Komponenten, die magnetische Felder produzieren, stark beeinträchtigt wurde. Daher wurden teilweise abweichende Werte geliefert, die nicht mehr in Abhängigkeit zum Norden standen. Generell konnten im Inneren eines Gebäudes Störungen beobachtet werden, die sich u. a. durch metallische Gegenstände in Baumaterialien erklären lassen. Auch bei der vorinstallierten Kompass-App der Google Glass konnten diese Störungen reproduziert werden. Sie waren bei der Google Glass im Vergleich mit dem Smartphone besonders signifikant. Dies könnte daher auch auf die eingebaute Hardware zurückzuführen sein. Da Platinen und Hardwarekomponenten eines mobilen Geräts, wie die Google Glass auch aus metallischen bzw. elektrischen Komponenten zusammengesetzt ist, sind Störungen aufgrund der Bauweise der Datenbrille nicht auszuschließen.
Eine entsprechende Kompasskalibrierung kann bei mobilen Geräten helfen, das Auftreten von ferromagnetischen Materialien, wie die oben genannten, zu kompensieren. Bei Smartphones empfiehlt man z. B. , das Gerät in Form einer acht zu rotieren. Da das Magnetfeld der Erde unabhängig von der Orientierung des Sensors relativ konstant ist, bewegen sich in der Theorie die Ausgabe-Werte des Magnetometers konstant auf einer Sphäre im dreidimensionalen Raum. In der Praxis entstehen jedoch durch die oben genannten Faktoren Ausreißer. Umso mehr das Gerät in alle drei Richtung bewegt wird, umso mehr Daten kann dieses sammeln, um die Ausreißer zu kompensieren und korrekt auszugeben. Google stellt allerdings hinsichtlich der Kalibrierung des Magnetometers für die Google Glass keine entsprechenden Informationen zur Verfügung. Zur Kalibrierung wurde daher die Datenbrille abgenommen und ähnlich wie bei einem Smartphone in Form einer Acht rotiert. Hierdurch konnten bereits signifikante Verbesserungen

erzielt und virtuelle Objekte exakter platziert werden.

Als nächstes wurde die Überlagerung der Realität und der Grad der Verschmelzung der virtuellen Objekte mit dieser evaluiert. Der wichtigste Faktor für die Verschmelzung von realer und virtueller Realität ist die Registrierung. Dabei spielen im Grunde drei Faktoren eine Rolle: Zum einen muss gewährleistet sein, dass die virtuellen Objekte im Raum korrekt eingefügt werden. Da im Kontext dieser Arbeit zweidimensionale Inhalte visualisiert werden, ist eine geometrische Registrierung, d.h. das Einfügen virtueller Inhalte in die dreidimensionale Welt schlicht nicht realisierbar. Ungeachtet dessen haben die Objekte dennoch einen direkten Bezug zur dreidimensionalen Welt, da die Objekte Kontext-abhängig, d.h. abhängig von der jeweiligen Position, stets an der gleichen Stelle visualisiert werden (z. B. Raum-Informationen vor einer Tür). Ferner werden Objekte in Abhängigkeit ihrer Entfernung entsprechend skaliert, wodurch die dritte Dimension nicht ungeachtet bleibt und für den Benutzer zumindest eine Tiefenwirkung entsteht.
Ein weiterer Faktor ist sicherlich die Antwortzeit des AR-Systems. In dem Zusammenhang fiel auf, dass bei schnellen Bewegungen des Kopfes minimale Schwimmeffekte, also Verzögerungen des virtuellen Bildes wahrgenommen werden. Diese zeitlichen Verzögerungen setzen sich aus der Antwortzeit des Sensors, also der Tracking-Rate und der Zeit zur Berechnung und Ausgabe der virtuellen Objekte zusammen. Selbst bei minimalen Verzögerungen im Millisekunden-Bereich, werden diese vom menschlichen Auge wahrgenommen. Wie bereits in den Grundlagen beschrieben, ist dies ein bekanntes Problem der Optical-See-Through Technik. Die Tracking-Rate ist zu stark von der Hardware abhängig, als dass man auf diese Softwareseitig Einfluss nehmen bzw. Optimierungen vornehmen könnte. Für die Verarbeitungszeit wurden bereits während der Konzeptphase entsprechende Optimierungen vorgenommen, wodurch die Verarbeitungszeit bis zur Darstellung der POIs reduziert werden konnte. Um die Antwortzeit des AR-Systems noch weiter zu reduzieren wären zudem spezielle Algorithmen, wie der Kalman-Filter denkbar, mit welchen man die Bewegungen vorausberechnen könnte[93].
Zudem wurden bei den virtuellen Objekten ein leichtes „zittern“ wahrgenommen, die als sehr störend empfunden worden sind. Diese entstanden durch die sehr empfindliche Intertialsensorik, die ein leichtes Messrauschen verursachte. Um das Messrauschen zu kompensieren wurde daher ein einfacher Tiefpassfilter eingesetzt. Der Filter erhält als Eingabe die Messwerte als Liste. Für jeden Messwert berechnet der Filter mithilfe eines Gewichtungsfaktors und des alten Werts einen neuen Wert. Der Gewichtungsfaktor

[93] Vgl. Tönnis, *Augmented Reality: Einblicke in die Erweiterte Realität*, S.83

entscheidet, wie stark der neue Wert den alten geglätteten Wert beeinflussen soll.
Der letzte Faktor, der den Grad der Verschmelzung beeinflusst, ist die Art der Darstellung und Ausgabe. Zur Realisierung der Optical-See-Through Augmented Reality war ein optischer Kombinierer, wie bspw. ein Prisma notwendig, weshalb die Google Glass schließlich zur Verwendung kam. Zudem wurde ein schwarzer Hintergrund gewählt, um die Transparenz gegenüber dem Hintergrund zu erreichen, und die Lichtintensität des Projektors reduziert, um die Konturen des virtuellen Bildes verblassen zu lassen und damit einen hohen Verschmelzungsgrad zu erreichen. In den Tests konnte man beobachten, dass dies bei sehr hellen Umgebungen (z. B. bei Tageslicht) sehr gut funktioniert, jedoch bei dunklen Umgebungen das Licht des Projektors immer noch zu hell ist, wodurch die Konturen sichtbar bleiben. Eine noch geringere Lichtintensität ist aufgrund der Einschränkung der Android API nicht realisierbar. Ungeachtet dessen, würde eine noch geringere Lichtintensität womöglich dazu führen, dass das Gesamtbild und damit auch die virtuellen Objekte noch blasser erscheinen. Hierbei muss berücksichtigt werden, dass umso blasser die virtuellen Objekte dargestellt werden, der Verschmelzungsgrad mit der Realität umso geringer ausfällt. Einen optimalen Kompromiss zwischen beiden Faktoren konnte schließlich nicht gefunden werden, wodurch der Verschmelzungsgrad um einiges einbüßt. Ferner stellte sich aus den Tests das Problem heraus, dass beim Verlassen der virtuellen Objekte aus dem Blickfeld der Google Glass, diese am Rand nur teilweise visualisiert werden, wogegen die reale Welt über das Display der Brille hinaus weiterhin sichtbar bleibt. Dies ist ein weiterer Effekt, wodurch der Realitätsgrad schwindet.

Zuletzt wurde in den Tests untersucht, in wie weit die Anforderungen einer Indoor-basierten AR-Navigation erfüllt wurden. Hierzu zählt sicherlich die Genauigkeit der Positionsbestimmung. Diese wurden bereits in den ersten Tests ausgiebig untersucht, wobei zum finalen Stand der Arbeit besonders die markerbasierte Methode eine hohe Genauigkeit versprach. Daher wurden die Tests ausschließlich auf diese Methode beschränkt. Schließlich galt es zu untersuchen, ob immer der optimale Pfad zum gewünschten Ziel berechnet wurde. Hierfür wurden immer für jeden Test zwei festgelegte Wegpunkte als Quelle und Ziel ausgewählt. Von einem bestimmten Startpunkt aus, wurde die ARApp ausgeführt und in den Navigationsmodus versetzt. Insgesamt kann man sagen, dass eine optimale Wegfindungsquote von 100% erreicht wurde. Dabei muss berücksichtigt werden, dass vorher optimale Wegpunkte und Laufwege im Mapeditor gesetzt und damit auch optimale Voraussetzungen geschaffen wurden. Die Qualität der Navigation hängt also stark von der zugrundeliegenden Karte ab. Zudem wurden Pfade

stets mit sehr geringer Gesamtlaufzeit berechnet, sodass der Nutzer nicht beeinträchtigt wird.
Ferner wurde die Art der Wegführung untersucht. Insgesamt war das Ergebnis zufriedenstellend. In allen Fällen wurde nach Bestimmung der jeweiligen Position der Weg zum nächsten Waypoint korrekt ausgegeben. In einigen Fällen, waren die Symbole jedoch versetzt, bedingt durch die oben genannten Störungen.

6 Fazit & Ausblick

Zum Schluss sollen nochmals die Kernpunkte der Arbeit zusammengefasst und ein Ausblick für mögliche Optimierungen gegeben werden. Ziel der Arbeit war es, den Nutzen der Google Glass im Alltag und ihr Potenzial als AR-Brille aufzuzeigen. Mit der Arbeit wurde ein ortsbasierter AR Ansatz vorgestellt, der innerhalb von Gebäuden zum Einsatz kommt. Als Beispielszenario diente die AR-basierte Navigation innerhalb des Gebäudes der TH Köln. Hierbei wurde ein Umgebungsmodell in Form eines Regular Grids konstruiert, welches eine diskrete Positionsbestimmung im Raum erlaubt. Ferner wurden verschiedene Verfahren zur Positionsbestimmung innerhalb von Gebäuden gegenübergestellt und dessen Vor- und Nachteile evaluiert. Hierbei wurden die Erkentnisse gewonnen, dass die Positionsbestimmung mittels LLA Markern die höchste Stabilität, jedoch die geringste Kontinuität aufweist, da Marker zunächst erkannt werden müssen, bevor die Position bestimmt werden kann.
Mit WLAN-Fingerprinting konnte eine Genauigkeit auf 1-2 Metern erreicht werden. Jedoch leidet die Positionserkennung nach wie vor unter Instabilität, die sich durch häufigen Positionswechsel und Springen der virtuellen Objekte bemerkbar macht. Es wurde deutlich, dass aufgrund der Mehrwegeausbreitungen der Funksignale eine hohe Streuung der Signalstärken vorliegt, die nur durch komplexere Verfahren zu kompensieren sind. Besonders die Instabilität, also die häufigen Positionswechsel beeinträchtigen die Qualität der ortsbasierten AR-Anwendung maßgeblich. Zur Optimierung des Systems ist also vor allem eine höhere Stabilität der aktuell ermittelten Position anzustreben. Dies gilt besonders dann, wenn der Benutzer an einer Stelle stehen bleibt und seine Position nicht ändert. Wünschenswert, wäre es jedoch in der Praxis auch, wenn kleinere Bewegungen kontinuierlich erkannt und die virtuellen Objekte entsprechend verschoben würden. Denn bisher wurde vorausgesetzt, dass das zugrundeliegende Umgebungsmodell ein diskretes Koordinatensystem ist, welches nur einen Positionswechsel von einem Feld zum anderen erlaubt. Zur sinnvollen Erweiterung dieser Arbeit ist also neben der Stabilität auch eine gewisse Kontinuität der Positionserkennung anzustreben. Dies muss vor allem unter Hinzunahme von weiteren Sensoren erfolgen, wodurch Schritte bzw. Bewegungen und die Blickrichtung erkannt und daraus ein Bewegungsmodell abgeleitet werden kann. Dieses Verfahren wird auch Dead Reckoning genannt. Um wiederum die ermittelte Position aus dem Fingerprinting mit der des Dead Reckoning Algorithmus zusammenzuführen, bieten sich sogenannte Partikel-Filter[94] an. Hier könnten mehrere wahrscheinliche Positionen, die

[94] Vgl. Reisig und Freytag, *Informatik - Aktuelle Themen im historischen Kontext*, S.42 ff.

aus dem Fingerprinting-Algorithmus ermittelt wurden, als sogenannte Partikel initialisiert werden. In dem Zeitintervall zwischen zwei Fingerprint-Messungen, könnte mittels Dead Reckoning, die geschätzte neue Position der Partikel ermittelt werden (Schätzung oder Vorhersage). Erhält man dann die nächste Fingerprint-Messung, wäre man in der Lage die Partikel anhand der Distanz zu den nächsten Fingerprint-Messungen zu gewichten (Korrektur). Ferner werden in jeder Phase Partikel mit einem Gewicht, das unter einem bestimmten Schwellwert liegt, eliminiert. Zudem werden Partikel, die über dem Schwellwert liegen erhöht, um damit auch die Wahrscheinlichkeitsdichte zu erhöhen. Zuletzt besteht die Möglichkeit des Map Matchings, bei welchem die Karteninformationen bei der Positionsbestimmung hinzugezogen werden.
Beim Bluetooth-Fingerprinting mittels iBeacons, konnte die Genauigkeit nur auf Kosten des Batterieverbrauchs verbessert werden. Die Verbesserung ist so minimal, dass sie dem Aufwand für häufigere Offline-Phasen aufgrund der Gesamtkapazität der Batterie nicht gerecht werden würde.
Auf Basis der Positionsbestimmung, wurde schließlich ein Konzept zur ortsbasierten AR mit der Google Glass vorgestellt. Hierbei wurde gezeigt, dass die Google Glass für Optical-See-Through AR die notwendigen Voraussetzungen bereitstellt und in der Lage ist, ortsbasierte, d.h. Kontext-abhängige Informationen in das Sichtfeld des Benutzers zu visualisieren. Durch die Einblendung von Belegungsinformationen der Räume im Sichtfeld des Betrachters und die Navigation von einer Position zur anderen, konnte einer der Hauptnutzen der AR, die Minimierung der Zeit zu Informationsbeschaffung, verdeutlicht werden.
Das Konzept wurde schließlich in Form einer geeigneten Software-Architektur entsprechend realisiert. Diese zeichnet sich durch offene Standards, hohe Verfügbarkeit und moderne Technologien aus. Zudem konnte die ortsbasierte AR im Indoor-Bereich erfolgreich mit der Google Glass getestet werden. Bei der Evaluation wurde jedoch deutlich, dass die AR-Applikation durch Faktoren wie Zweidimensionalität, Schwimmeffekte aufgrund der Verarbeitungszeit, Zittern, Verblassen und Überblenden der Objekte an Verschmelzungsgrad einbüßt. Einige dieser Faktoren sind optimierbar, andere durch die technischen Gegebenheiten bedingt. Schlussendlich konnte jedoch das Potenzial der Google Glass als AR-Brille verdeutlicht werden. Eine denkbare Verbesserung zur Erhöhung des Verschmelzungsgrades wäre etwa eine dreidimensionale Konstruktion der Karte, wodurch sich virtuelle Objekte durch die genauen Tiefeninformationen noch realistischer in die Realität einfügen würden und die Positionsbestimmung auf mehrere Etagen einfacher zu realisieren wäre. Zudem wären Filtermechanismen zur Vorhersage der Blickrichtung sinnvoll, um Schwimmeffekte reduzieren zu können. Ferner könnte man die Lichtintensität des Projektors je nach Helligkeit anpas-

sen, um optimale Lichtverhältnisse zu erhalten.
Die vorliegende Arbeit hat lediglich mit einem speziellen Anwendungsszenario den Nutzen und das Potenzial der Google Glass und Augmented Reality im Alltag aufgezeigt. Hingegen ist die Lösung so generisch, dass sie auch für andere Anwendungsbereiche anwendbar wäre. Beispielweise könnte die AR-App für die Lagerhaltung in der Industrie eingesetzt werden. So könnten etwa Stapelfahrer schneller mithilfe der Navigation zu den jeweiligen Waren gelangen und Informationen zu Waren in das Sichtfeld eingeblendet bekommen. In einer Bibliothek könnten Besucher schneller zum Bücherregal geführt werden, um ein bestimmtes Thema oder gar ein bestimmtes Buch zu finden. Ähnlich könnte man in einem Museum Informationen zu Objekten in das Sichtfeld des Benutzers einblenden. Je nach Anwendungsfall kann eine präzisere Positionsbestimmung notwendig sein. Hierfür sind sicherlich noch die genannten Optimierungen notwendig. Der Nutzen einer solchen Anwendung ist jedoch vielfältig.

Literatur

Azuma, Ronald T.: A survey of augmented reality. Presence: Teleoperators and Virtual Environments, 6 August 1997 Nr. 4, 355–385

Baggio, Daniel Lélis et al.: Mastering OpenCV with Practical Computer Vision Projects -. Birmingham: Packt Publishing Ltd, 2012

Bahl, Paramvir und Padmanabhan, Venkata N.: RADAR: an in-building RF-based user location and tracking system. 2000, 775–784

Bimber, Oliver und Raskar, Ramesh: Spatial Augmented Reality: Merging Real and Virtual Worlds: A Modern Approach to Augmented Reality. A K Peters Ltd (Ma), 7 2005 ⟨URL: http://amazon.de/o/ASIN/1568812302/⟩, 392 Seiten

Bormann, S.: Virtuelle Realität: Genese und Evaluation. Addison-Wesley, 1994 ⟨URL: http://books.google.de/books?id=ske7AgAACAAJ⟩

Brin, Sergey: Sergey Brin: Why Google Glass? o. O., Februar 2013 ⟨URL: https://www.ted.com/talks/sergey_brin_why_google_glass⟩ – Zugriff am 09.09.2015

Caudell, T. P. und Mizell, D. W.: Augmented Reality: An Application of Heads-Up Display Technology to Manual Manufacturing Processes. In International Conference on System Sciences. 1992 ⟨URL: http://ieeexplore.ieee.org.arugula.cc.columbia.edu:2048/stamp/stamp.jsp?arnumber=183317\&isnumber=4717⟩, 73–80

Chen, Lina et al.: An Improved Algorithm to Generate a Wi-Fi Fingerprint Database for Indoor Positioning. Sensors, 13 2013 Nr. 8, 11085–11096 ⟨URL: http://www.mdpi.com/1424-8220/13/8/11085⟩

Cormen, Thomas H et al.: Algorithmen - Eine Einführung -. überarbeitete und aktualisierte Auflage Auflage. MÃ¼nchen: Oldenbourg Verlag, 2010

Dorner, Ralf (Hrsg.): Virtual und Augmented Reality (VR/AR): Grundlagen und Methoden der Virtuellen und Augmentierten Realität (eXamen.press) (German Edition). 2013. Auflage. Springer Vieweg, 2 2014 ⟨URL: http://amazon.de/o/ASIN/3642289029/⟩, 368 Seiten

Dr. Weissman, Zeev: Indoor Location.

Firstenberg, Allen und Salas, Jason: Designing and Developing for Google Glass - Thinking Differently for a New Platform. 1. Auflage. Sebastopol, California: O'Reilly Media, Incorporated, 2014

Fischer G., Dietrich B., Winkler F.: Bluetooth Indoor Localization System. o. O., o. J.

Furht, Borko (Hrsg.): Handbook of Augmented Reality. Springer, 2011 ⟨URL: http://dblp.uni-trier.de/db/books/daglib/0027797.html⟩

Gamma, Erich et al.: Design Patterns - Elements of Reusable Object-Oriented Software. 1. Auflage. Amsterdam: Pearson Education, 1994

Izquierdo F., Ciurana M. Barceló F., Paradells J. und E., Zola: Performance evaluation of a TOA-based trilateration method to locate terminals in WLAN. o. O., 2006

K., Curran et al.: An Evaluation of Indoor Location Determination Technologies. Juni 2011

Kawaji, Hisato et al.: Image-based Indoor Positioning System: Fast Image Matching Using Omnidirectional Panoramic Images. In Proceedings of the 1st ACM International Workshop on Multimodal Pervasive Video Analysis. New York, NY, USA: ACM, 2010, MPVA '10 ⟨URL: http://doi.acm.org/10.1145/1878039.1878041⟩, 1–4

King, Thomas et al.: COMPASS: A Probabilistic Indoor Positioning System Based on 802.11 and Digital Compasses. In Proceedings of the 1st International Workshop on Wireless Network Testbeds, Experimental Evaluation & Characterization. New York, NY, USA: ACM, 2006, WiNTECH '06 ⟨URL: http://doi.acm.org/10.1145/1160987.1160995⟩, 34–40

Mehler-Bicher, A., Reiß, M. und Steiger, L.: Augmented Reality: Theorie und Praxis. Oldenbourg Wissenschaftsverlag, 2011 ⟨URL: http://books.google.de/books?id=kmzgXVYXjCkC⟩

metaio: Indoor Location Based Channels. o. O., 2015 ⟨URL: https://dev.metaio.com/junaio/quickstarts/location-based-quickstarts/indoor-location-based-channels/⟩ – Zugriff am 08.08.2015

Milgram, P. und Kishino, F.: A Taxonomy of Mixed Reality Visual Displays. IEICE Trans. Information Systems, E77–D Dezember 1994 Nr. 12, 1321–1329 ⟨URL: "http://vered.rose.utoronto.ca/people/paul_dir/IEICE94/ieice.html"⟩

o. V.: ARUCO How to create markers | iplimage. o. O., März 2012 ⟨URL: http://iplimage.com/blog/create-markers-aruco/⟩ – Zugriff am 05.09.2015

o. V.: 3DReco Homepage. o. O., o. J. ⟨URL: http://www.mi.hs-rm.de/~schwan/Projects/CG/CarreraCV/doku/intrinsisch/intrinsisch.htm⟩ – Zugriff am 08.08.2015

o. V.: Active Badge | Indoor-Ortung.de. o. O., o. J. ⟨URL: http://indoor-ortung.de/systeme/active-badge/⟩ – Zugriff am 02.08.2015

o. V.: Active Bat | Indoor-Ortung.de. o. O., o. J. ⟨URL: http://indoor-ortung.de/systeme/active-bat/⟩ – Zugriff am 03.08.2015

o. V.: Adding Real-world Context with Estimote Beacons and Stickers | Xamarin Blog. o. O., o. J. ⟨URL: https://blog.xamarin.com/adding-real-world-context-with-estimote-beacons-and-stickers/⟩ – Zugriff am 09.09.2015

o. V.: CS-9645 Introduction to Computer Vision Techniques Winter 2014. o. O., o. J. ⟨URL: http://www.csd.uwo.ca/faculty/beau/CS9645/CS9645-Feature-Detection-and-Matching.pdf⟩ – Zugriff am 03.08.2015

o. V.: Google Developers. o. O., o. J. ⟨URL: https://developers.google.com/glass/design/ui⟩ – Zugriff am 14.08.2015

o. V.: Google Developers. o. O., o. J. ⟨URL: https://developers.google.com/glass/develop/overview⟩ – Zugriff am 19.08.2015

o. V.: Gypsy Motion capture system technology explained. o. O., o. J. ⟨URL: http://www.metamotion.com/gypsy/gypsy-motion-capture-system-mocap.htm⟩ – Zugriff am 27.07.2015

o. V.: iBeacon for Developers - Apple Developer. o. O., o. J. ⟨URL: https://developer.apple.com/ibeacon/⟩ – Zugriff am 30.08.2015

o. V.: Locations and Sensors | Glass | Google Developers. o. O., o. J. ⟨URL: https://developers.google.com/glass/develop/gdk/location-sensors⟩ – Zugriff am 20.08.2015

o. V.: SensorManager | Android Developers. o. O., o. J. ⟨URL: http://developer.android.com/reference/android/hardware/SensorManager.html⟩ – Zugriff am 20.08.2015

o. V.: Vektorgrafik, Verschlusszeit, Verzeichnung, Vignettierung, Vollautomatischer Weißabgleich bei Digitalkameras, Vollformatsensor. o. O., o. J. ⟨URL: http://www.filmscanner.info/Glossar_V.html⟩ – Zugriff am 08.08.2015

o. V.: Virtual Reality Tracking Systems - HowStuffWorks. o. O., o. J. ⟨URL: http://electronics.howstuffworks.com/gadgets/other-gadgets/VR-gear6.htm⟩ – Zugriff am 30.08.2015

o. V.: What are Broadcasting Power, RSSI and other characteristics of beacon's signal? – Estimote Community Portal. o. O., o. J. ⟨URL: https://community.estimote.com/hc/en-us/articles/201636913-What-are-Broadcasting-Power-RSSI-and-other-characteristics-of-beacon-s-signal-⟩ – Zugriff am 06.08.2015

Popovic, Dragan: Grundlagen der drahtlosen Übertragung. Fachhochschule Wedel – Informatikseminar WS2004, 2004 ⟨URL: http://www.fh-wedel.de/archiv/iw/Lehrveranstaltungen/WS2004/SeminarMC/Ausarbeitung1PopovicGrundlagen.pdf⟩

Reisig, Wolfgang und Freytag, Johann-Christoph: Informatik - Aktuelle Themen im historischen Kontext. Berlin Heidelberg New York: Springer-Verlag, 2006

Remus, Jens: Informatik Seminar Spiele-KI - Ermittelung eines guten Wege-netzes für Navigationsalgorithmen. Fachhochschule Wedel – Informatikseminar SS2007, 2007 ⟨URL: http://de.scribd.com/doc/490678/Ermittlung-eines-guten-Wegenetzes-fur-Navigationsalgorithmen-Ausarbeitung⟩

Rose M., Fehling D.: Augmented Reality und Printmedien. o. O., Januar 2015 ⟨URL: http://www2.tu-ilmenau.de/zsmp/Augmented_Reality_Rose_Fehling⟩ – Zugriff am 20.08.2015

Scheinmann, Jonathan: Augmented Reality - Definition, Realisierung und Nutzen. Juni 2014

Schumann, M., Achilles, S. und Müller, S.: Analysis by Synthesis Techniques for Markerless Tracking. In Analysis by Synthesis Techniques for Markerless Tracking. Virtuelle und Erweiterte Realität, 6. Workshop der GI Fachgruppe VR/AR o. O., 2009

Siimon, Reynolds: Why Google Glass Failed: A Marketing Lesson. o. O., Februar 2015 ⟨URL: http://www.forbes.com/sites/siimonreynolds/2015/02/05/why-google-glass-failed/2/⟩ – Zugriff am 05.08.2015

Suthau, Tim: Augmented Reality - Positionsgenaue Einblendung räumlicher Informationen in einem See Through Head Mounted Display für die Medizin am Beispiel der Leberchirurgie. Dissertation, o. O., 2006

Sutherland, Ivan E.: A Head-mounted Three Dimensional Display. In Proceedings of the December 9-11, 1968, Fall Joint Computer Conference, Part I. New York, NY, USA: ACM, 1968, AFIPS '68 (Fall, part I) ⟨URL: http://doi.acm.org/10.1145/1476589.1476686⟩, 757–764

Tönnis, Marcus: Augmented Reality. Springer, 2010, Informatik im Fokus ⟨URL: http://books.google.de/books?id=XMvEB9jQ_28C⟩

Tönnis, Marcus: Augmented Reality: Einblicke in die Erweiterte Realität. Springer-Verlag New York Inc, 2010

Vision, Wear: Google Glass - Was ist das eigentlich? o. O., 2015 ⟨URL: http://www.wearvision.de/googleglass/⟩ – Zugriff am 01.08.2015

Volpe, J.: Sergey Brin demos Project Glass onstage at Google I/O. o. O., o. J. ⟨URL: http://www.engadget.com/2012/06/27/sergey-brin-demos-project-glass-on-stage-at-google-i-o/⟩ – Zugriff am 20.08.2015

Youssef, Moustafa und Agrawala, Ashok: The Horus WLAN Location Determination System. In Proceedings of the 3rd International Conference on Mobile Systems, Applications, and Services. New York, NY, USA: ACM, 2005, MobiSys '05 ⟨URL: http://doi.acm.org/10.1145/1067170.1067193⟩, 205–218

Anhang

Aktualisierung der RadioMap-Tabelle

```
1
2  SET @N = (SELECT COUNT(*) n FROM Beacon WHERE
       idField=311 and  idAP=2 );
3  SET @lambda =
       0.00000000000000000000000000000000000000001;
4  INSERT INTO RadioMap (idField,idAP,signalstrength,
       frequency,probability)
5
6  (SELECT s.idField,s.idAP,s.signalstrength,
       orientation,IFNULL(frequency,0),IFNULL(
       probability+@beta,@lambda) FROM
7  (SELECT idField,idAP,signalstrength,orientation,
       COUNT(signalstrength) AS frequency,COUNT(
       signalstrength)/@N AS probability FROM Beacon
       WHERE idField=311 AND idAP=2
8  GROUP BY idField,idAP,signalstrength,orientation) b
9  RIGHT JOIN
10 (SELECT 311 as idField,2 as idAP, signalstrength
       FROM Space) s ON s.signalstrength = b.
       signalstrength AND b.idField=s.idField AND b.
       idAP=s.idAP)
11
12 ON DUPLICATE KEY UPDATE frequency = VALUES(
       frequency),probability = VALUES(probability);
```

Aggregation von Einzelwahrscheinlichkeiten

```
1  SELECT r.x,r.y,exp(sum(log(coalesce(r.probability,1
       e-20))))
2  FROM (
3
4  (SELECT f.x,f.y,r.probability FROM fhkoelnar.Field
       f LEFT JOIN fhkoelnar.RadioMap r ON f.x = r.x
       AND f.y = r.y AND r.signalstrength=-71.0 AND r.
       macaddress='EC:9E:3A:D6:60:78' AND abs((
       orientation + 180 -  35.332) \% 360 - 180) <=
       70)
5
```

```
union all

(SELECT f.x,f.y,r.probability FROM fhkoelnar.Field
   f LEFT JOIN fhkoelnar.RadioMap r ON f.x = r.x
   AND f.y = r.y AND r.signalstrength=-74.0 AND r.
   macaddress='D5:26:4C:E9:F4:2A' AND abs((
   orientation + 180 -  35.332) \% 360 - 180) <=
   70)

union all
...
)
r GROUP BY r.x,r.y
```

JSON Objekt vom TH-Köln Stundenplan Webservice

```
[
{"fbnr":"20",
"bezeichnung":"Oberseminar für Doktoranden",
"kurzbez":"OB",
"pruefung_typ":"FP",
"raum_kz":"2112",
"raum_bezeichnung":"PC-Pool Mathe für Ingenieure",
"fachkuerzel":"OB",
"semester_nr":"1",
"fach_typ":"V",
"wochentag":"2",
"zeit":"10",
"sg_kz":"ZV",
"zeit_string":
"17.00-17.45",
"dozent":"BI Thomas Bartz-Beielstein",
"wochentag_string":"Dienstag"
},
{...},
...
]
```

Selbstständigkeitserklärung

Ich versichere an Eides Statt, die von mir vorgelegte Arbeit selbständig verfasst zu haben. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben. Die Arbeit hat mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegen.

Ratingen, den xx.xx.xxxx